Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Mitja Golob

IoT v stavbah ob uporabi SNMP

diplomsko delo
na visokošolskem strokovnem študiju

Mentor: dr. Andrej Brodnik

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Takoimenovani inteligentni objekti so primer nečesa, čemur danes rečemo z eno besedo Internet stvari ali IoT. Pri IoT imamo tri ključne dejavnike: senzorje, prožilnike (aktuatorje) in komunikacijo. V nalogi se osredotočite na senzorje in komunikacijo ter možnost uporabe za slednjo preprostega SNMP protokola. Preučite možnost uporabe tako cenene strojne opreme kot odprtokodne implementacije agentov in storitev slonečih na protokolu SNMP.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Mitja Golob, z vpisno številko 63010029, sem avtor diplomskega dela z naslovom:

IoT v stavbah ob uporabi SNMP

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom dr. Andreja Brodnika
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki „Dela FRI“.

V Ljubljani, dne 9. september 2016 Podpis avtorja:

Lepo se zahvaljujem mentorju doc. dr. Andreju Brodniku in as. dr. Gašperju Feletu Žoržu za idejo, strokovno pomoč in vodenje pri nastajanju diplomskega dela. Dragi starši, sestra in brat, hvala za spodbujanje v času študija in potrpežljivo čakanje na zagovor diplome. Zahvala gre tudi Armandu Kovačiču za vse ideje in debate. Posebna zahvala gre tebi, draga Andreja, za vso podporo, animacijo in prijazno pomoč.

Kazalo

Seznam uporabljenih kratic

BASH	Bourne Again Shell
BER	Basic Encoding Rules
CPE	Centralno procesna enota
DES	Data Encryption Standard
DOS	Denial Of Service
DS	Data Source
IP	Internet Protocol
LAMP	Linux Apache Mysql PHP
MIB	Management Information Base
NPCD	Nagios-Perfdata-C-Daemon
OID	Object Identifier
PDU	Protocol Data Unit
RRD	Round-Robin Database
RRDtool	Round-Robin Database Tool
SCADA	Supervisory Control And Data Acquisition
SD	Secure Digital
SMS	Short Message Service
SNMP	Simple Network Management Protocol
UDP	User Datagram Protocol
USM	User-Based Security Model
VACM	View-Based Securtiy Model
XML	Extensible Markup Language

Povzetek

Večina srednjih in večjih podjetij se srečuje s potrebami zajema podatkov o stanju temperature v sistemskih prostorih s strežniško in mrežno opremo. Skupno število takih prostorov hitro preseže število 50 ali več, kar pomeni visoke stroške za nakup strojne in programske opreme. Cilj diplomskega dela je bil najti preprosto in poceni rešitev za namestitev nadzornega sistema, ki samodejno meri in prikazuje stanje temperature v različnih prostorih. Naša rešitev je zasnovana na strojni opremi Raspberry Pi, protokolu **SNMP** in nadzornem sistemu **Nagios Core**. Končni rezultat je prikaz stanja temperature več naprav, prav tako sistem omogoča vpogled v zgodovino stanj in obveščanje uporabnikov po elektronski pošti.

Ključne besede: internet stvari, SNMP, Nagios, Net-SNMP, MIB, OID, PNP4Nagios.

Abstract

In most medium and large companies, the monitoring of temperature in rooms containing servers and networking equipment is required. The number of rooms can quickly reach 50 or more, which means a significant funds must be spent on monitoring hardware and software. The goal of this diploma thesis was to implement a straight-forward and cost-effective IT infrastructure monitoring system for automatic surveying and display of temperature in different rooms. The hardware part of our solution is based on the Rasperry Pi while the software part uses the **SNMP** for communication and **Nagios Core** for monitoring. The final result allows a user to view the history and current temperature for multiple devices. The system also supports the notification of users via e-mail.

Title: IoT in Buildings using SNMP

Keywords: internet of things, SNMP, Nagios, Net-SNMP, MIB, PNP4Nagios, OID.

Poglavje 1

Uvod

Sodobna oprema nam za nizko ceno omogoča neprekinjeno spremljanje pogojev v prostoru kot so, temperatura, relativna zračna vlažnost, prisotnost dima ali tekočin itd.

Na tržišču obstaja veliko ponudnikov, ki ponujajo svoje rešitve po zelo visokih cenah. Storitev lahko kupimo v paketu, pri čemer hkrati dobimo programsko rešitev in ustrezno strojno opremo. Glavna slabost takih paketov je, da smo vezani na določenega ponudnika, vsaka potreba po dodatnem senzorju ali nadgradnji aplikacije za podjetje predstavlja dodaten strošek. Lahko se odločimo za nakup programske opreme pri enem ponudniku, medtem ko strojno opremo kupimo pri drugem. Tu se srečamo s težavami združljivosti različne opreme. Ker gre za zaprtokodne licenčne programe, je potrebno za pravilno komunikacijo med napravo in programom izvesti spremembo programske kode, kar zopet predstavlja dodatne stroške.

S takšno težavo smo se srečali v enem od slovenskih podjetij. Zaradi varnostnih vidikov je prišlo do zahteve po spremljanju in beleženju temperature v vseh sistemskih prostorih s strežniško in mrežno opremo. V velikem podjetju se seštevek takih prostorov hitro približa številki 50 ali več. Cena strojne oprema, ki jo moramo namestiti v vsak prostor, lahko presega vrednost 500 eurov na kos. Poleg tega vsaka nadaljna nadgradnja funkcionalnosti pomeni dodatno stroškovno obremenitev. Licenčnina za programsko opremo lahko

presega tudi 5 mestno številko. Nemalokrat se zgodi, da v ohišju strojne opreme ponudnikov domuje kar ena od široko uporabljenih rešitev kot je Arduino, BeagleBone, Raspberry Pi, Nanode, Waspmote.

Kot dokaz koncepta (angl. *proof of concept*) smo postavili rešitev, ki je zasnovana tako na strojni kot programski opremi. Glavni kriteriji za sprejemljivost končne rešitve so bili:

- nizka cena in razpoložljivost strojne opreme na trgu,
- zanesljiva in poceni rešitev na ravni programske opreme,
- možnost preprostega dodajanja nove opreme in menjave opreme v primeru okvare,
- preprosta administracija,
- obveščanje naslovnikov v primeru mejnih odstopanj po elektronski pošti,
- možnost razširljivosti sistema z dodatnimi nadzornimi senzorji in
- grafični prikaz podatkov in zgodovine vrednosti stanj.

1.1 Struktura diplomskega dela

V uvodnem delu diplomskega dela smo opisali problematiko in postavili kriterije, na katerih je zasnovana naša rešitev. Osrednji del opisuje protokol **SNMP**, že obstoječi programski rešitvi (Alarm, PcVue) ter programsko in strojno opremo, ki smo jo uporabili. Zatem smo opisali infrastrukturo, ki je zasnovana na strojni opremi Raspberry Pi, protokolu **SNMP** in nadzornem sistemu **Nagios Core**. V zaključnem delu diplomskega dela smo predstavili prednosti in slabosti predlagane rešitve, primerjali našo rešitev z že obstoječima rešitvama in izpostavili, zakaj je bila naša izbira najbolj primerna.

Poglavje 2

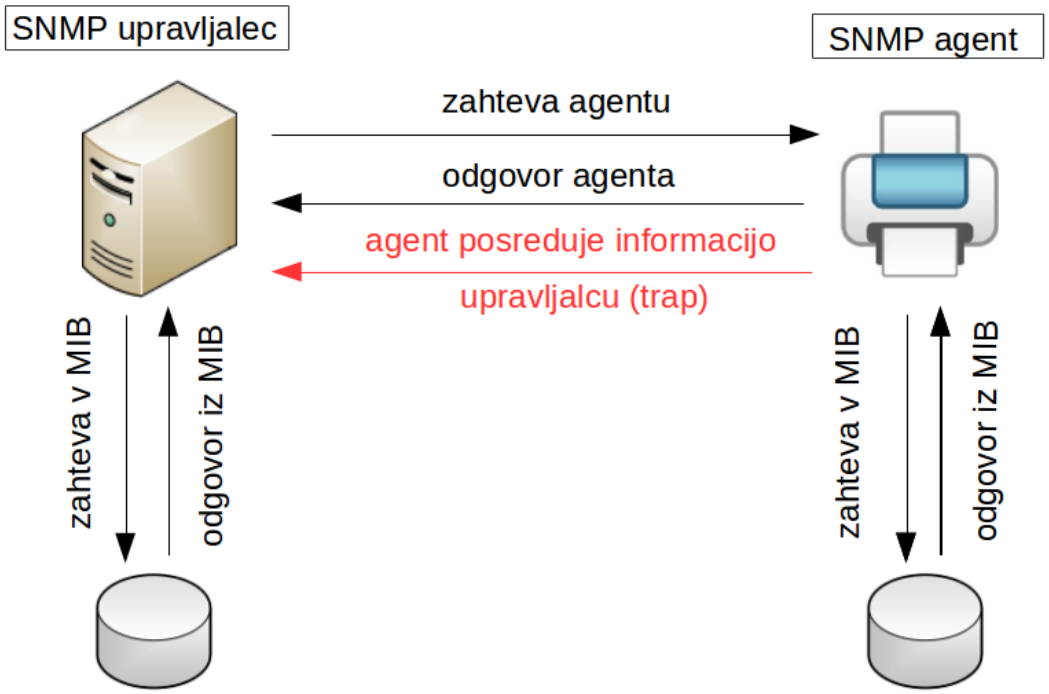
Protokol SNMP

SNMP (angl. *Simple network management protocol*) je standardni internetni protokol za nadzor in upravljanje naprav povezanih v računalniško omrežje. Nadzorovane naprave in storitve so lahko mrežni usmerjevalniki, mrežna stikala, tiskalniki, računalniki, strežniki itd. Za uspešno komunikacijo po protokolu **SNMP** sta potrebni dve komponenti: upravljalec (angl. *manager*) in agent (angl. *agent*) [1]. Upravljalec je programska oprema za nadzorovanje in upravljanje naprav, ki pridobiva informacije o stanju naprave, nastavlja parametre naprave ter shranjuje in obdeluje pridobljene podatke. Agent je programska oprema, ki je nameščena na napravi, ki jo želimo nadzorovati. Naloga agenta je, da upravljalcu pošilja zahtevane podatke, katerih pomen je opisan v bazi **MIB** (Slika 2.1), lahko pa glede na preddoločene dogodke tudi sam obvešča upravljalca o stanju naprave (angl. *trap*). Prikaz delovanja protokola **SNMP** je predstavljen na Sliki 2.1.

SNMP za izmenjavo podatkov med upravljalcem in agentom uporablja transportni protokol **UDP** (angl. *User Datagram Protocol*). Za ta namen uporablja vrata (angl. *port*):

- 161 za pošiljanje in sprejemanje zahtev,
- 162 za sprejemanje obvestil s strani agenta (angl. *trap*).

Protokol **UDP** je nezanesljiv, saj v primeru izgube poslanega paketa po-



Slika 2.1: Prikaz delovanja protokola **SNMP**.

šiljatelj o tem ni obveščen. To pomanjkljivost ponavadi rešujemo tako, da nadzorni sistem po poslani zahtevi vklopi časovno kontrolo. Če po iztečenem času sistem ne prejme odgovora s strani agenta, zahtevo pošlje ponovno. Ta pomanjkljivost postane problematična v primeru, ko agent pošlje sporočilo *„trap“*. Če sporočilo ne prispe do nadzornega upravitelja, agent o tem ni obveščen in ne ve, da bi moral poslati sporočilo ponovno (ta pomankljivost je bila odpravljena v višji verziji protokola **SNMPv2c** z ukazom *„inform“*). Prednost uporabe protokola **UDP** je manjša obremenjenost računalniškega omrežja in s tem večja hitrost komunikacije, saj je v primerjavi s protokolom **TCP** (angl. *Transmision Control Protocol*), količina komunikacijskih paketov precej manjša [2].

Upravljalec in agent si izmenjujeta 5 vrst sporočil v obliki enot protokolnih podatkov **PDU** (angl. *protocol data unit*) [3]:

GetRequest: upravljalec od agenta zahteva pridobitev vrednosti objekta

v bazi **MIB**. Agent posreduje vrednost upravljalcu s sporočilom GetResponse.

GetNextRequest: upravljalec od agenta zahteva pridobitev naslednje vrednosti objekta iz baze **MIB**. Agent odgovori s sporočilom GetResponse.

SetRequest: upravljalec zahteva spremembo vrednosti določenega objekta v bazi **MIB**.

GetResponse: s sporočilom GetResponse agent posreduje vrednost ali odgovori, da je uspešno nastavil novo vrednost objekta. Agent s tem sporočilom odgovarja na zahteve upravljalca GetRequest, GetNextRequest, SetRequest.

Trap: agent ob določenih dogodkih posreduje sporočilo upravljalcu. V bazi **MIB** sta definirana cilj in naslov obvestila.

Kasneje sta bili v verziji **SNMPv2c** dodani še dve operaciji [3]:

InformRequest: potrditev prejetega trap sporočila s strani upravljalca, kar doda funkcionalnost, ki ne obstaja v transportnem protokolu **UDP**.

GetBulkRequest: zamenjava za ukaz GetNextRequest, kar omogoča lažjo poizvedbo po več vrednostih hkrati z enim ukazom.

Izmenjava sporočil **PDU** med upravljalcem in agentom je vidna na Sliki 2.2.

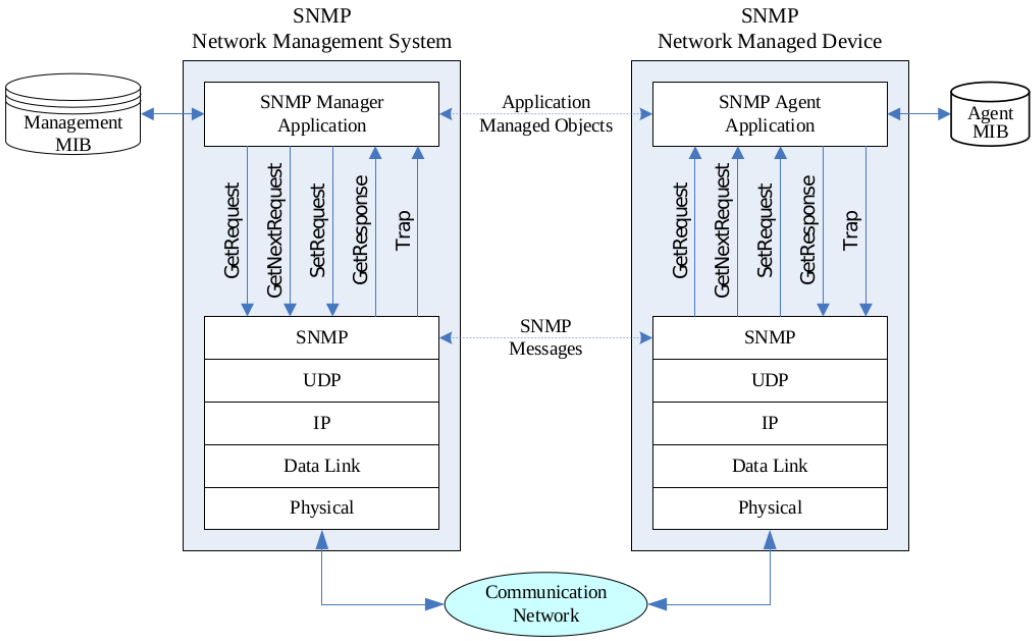
2.1 MIB, OID, SMI

MIB je kratica za bazo upravljalskih informacij (angl. *management identifier base*). Gre za hierarhično urejeno podatkovno bazo, v kateri so shranjeni opisi podatkov o posameznih objektih naprave. **SMI** (angl. *structure of management information*) je jezik, s katerim definiramo zapis objektov in njihovih sledečih parametrov [5]:

- poimenovanje objektov: vsak objekt ima definirano enolično ime. **SMI** dovoljuje poimenovanje v obliki imen (iso.org.dod.internet.mgmt) ali v obliki številčnih identifikatorjev objektov **OID** (1.3.6.1.2);

- tip objekta: **SMI** dovoljuje uporabo standardnih tipov kot so cela števila (integer), osmiška števila (octet), niz (string), kakor tudi nestandardne specifične tipe kot je npr. IP številka;

- metoda kodiranja objektov: **SMI** uporablja preprosto kodiranje imenovano **BER** (angl. *basic encoding rules*).



Slika 2.2: Slika prikazuje izmenjavo **PDU** sporočil med upravljalcem in agentom [3].

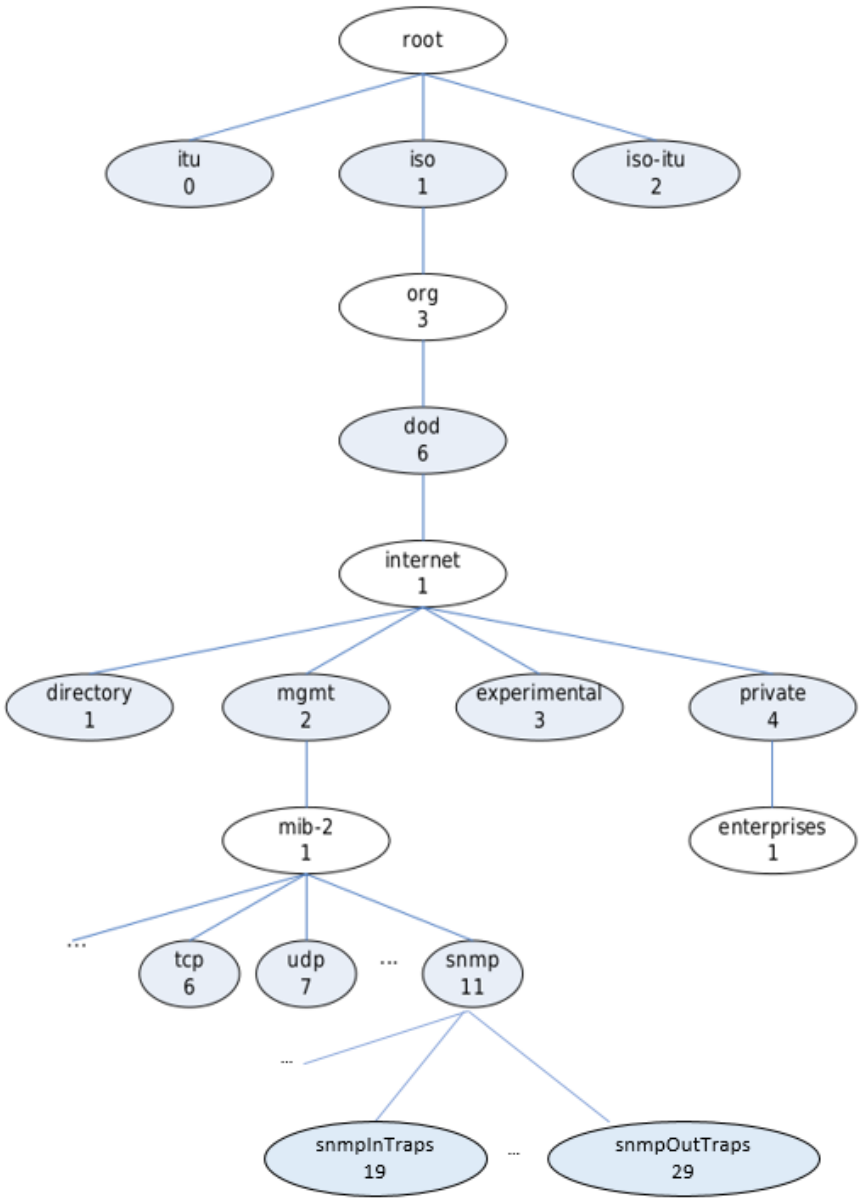
Primer definicije objekta *snmpInTraps* z jezikom **SMI**:

```
snmpInTraps OBJECT TYPE
      SYNTAX     Counter
      ACCESS     read-only
      STATUS     mandatory
      DESCRIPTION
       "The total number of SNMP Trap PDUs which have
       been accepted and processed by the SNMP protocol
       entity."
       ::= snmp 19
```

Seznam vseh objektov za nadzorovane naprave je shranjen v podatkovni bazi **MIB**. Do teh objektov dostopamo preko identifikatorjev objektov **OID** (angl. *object identifier*), ki so shranjeni v obliki drevesne strukture. Vrh takšne strukture se imenuje koren, nasledniki so podlisti, končnim objektom pa pravimo listi [4]. Kazalec **OID** na objekt je pot do objekta v drevesni strukturi. Če želimo najti opis objekta *snmpInTraps* v bazi **MIB** (opis **SMI** je prikazan v zgornjem okviru), lahko le-tega predstavimo s številko 1.3.6.1.2.1.11.19 (Slika 2.3). **OID** lahko predstavimo tudi v obliki imena. V tem primeru je zapis *iso.org.dod.internet.mgmt.mib-2.snmp.snmpInTraps* enak numeričnemu zapisu, pri čemer je iso=1, org=3, dod=6, intenet=1, mgmt=2, mib-2=1, snmp=11, snmpInTraps=19.

2.2 Varnostni vidik protokola SNMP

Prva verzija protokola **SNMPv1** je bila izdana leta 1988. V času začetkov razvoja protokola varnost ni bila na prvem mestu in zato v verziji **SNMPv1** kot tudi v verziji **SNMPv2c** vsa komunikacija med agentom in upravljalcem poteka v nekriptiranem načinu. Varnostni mehanizem je v tem primeru definicija niza skupnosti (angl. *community*). Poizvedba tako poleg same zahteve pošlje tudi geslo skupnosti v osnovnem tekstovnem načinu. Vsakdo, ki prestreže promet med upravljalcem in agentom, lahko na lahek način prido-



Slika 2.3: Prikaz drevesne stukture baze **MIB** [3].

bil dostop do podatkov. Drugi varnostni mehanizem, ki ga omogočata obe verziji, je nastavitev pravic dostopa do podatkov **MIB**. Tako lahko določeni upravljalci v skupnosti spreminjajo parametre v bazi **MIB** ali jih zgolj berejo. **SNMP** protokol lahko predstavlja resno varnostno grožnjo ne le za

napravo, ampak tudi za celotno informacijsko infrastrukturo.

Nadzorovana informacija je pri komunikaciji med upravljalcem in agentom izpostavljena različnim neavtoriziranim posegom [2]:

Avtentičnost sporočil: neavtoriziran uporabnik med potjo spremeni vsebino sporočila (podatkov), pri čemer izvorni in ponorni naslov ni spremenjen.

Zakrivanje podatkov: napadalec s spremembo originalnega naslova doseže, da se naslovniku predstavi kot avtoriziran uporabnik.

Zavrnitev storitve (DoS): napadalec blokira tok sporočil med upravljalcem in agentom.

Prestrezanje in prisluškovanje upravljanju omrežja.

Nekateri znani primeri varnostnih lukenj protokla **SNMP** so [7]:

- sprememba privzetega niza skupnosti: za vsako napravo je privzet niz skupnosti nastavljen na public. Če osnovnega niza ne popravimo, lahko vsakdo pridobi podatke o napravi;
- skupnost sploh ni nastavljena;
- nekateri HP tiskalniki omogočajo izvedbo napadov zavrnitve storitve DOS (angl. *denial of service*) z ukazi SNMPget. Na ta način lahko preko tiskalnikov napadalec povzroči izpad strežnikov, mrežne opreme itd.;
- nadzor dostopa: vsem skupnostim, ki imajo pravice pisanja, je potrebno omejiti dostop, saj v nasprotnem primeru napadalec lahko spreminja nastavitve naprave;

Kot odgovor na varnostne pomankljivosti verzije **SNMPv1** in **SNMPv2c**, je bila leta 1999 izdana verzija **SNMPv3**. Za ta namen sta bila implementirana dva varnostna modela:

- User-Based Security Model (**USM**) in
- View-Base Access Control Model (**VACM**).

2.3 Varnostni model USM

USM uporablja dva varnostna mehanizma, avtentikacijo in zasebnost z uporabo šifriranja. V primeru avtentikacije mehanizem zagotovi, da je bilo sporočilo poslano s strani pravega pošiljatelja in da sporočilo med pošiljanjem ni bilo spremenjeno. Za to je potrebno na strani pošiljatelja in prejemnika definirati avtentikacijski ključ (*authKey*). Pošiljatelj vključi avtentikacijsko kodo v sporočilo **SNMPv3**, prejemnik pa na drugi strani uporabi enak ključ, s katerim tudi sam izračuna avtentikacijsko kodo. Če prejemnik ugotovi, da je vrednost kode enaka vrednosti, ki je priložena prejetemu sporočilu, potem je jasno, da je sporočilo prišlo od pravega pošiljatelja. Drugi vidik mehanizma avtentikacije je preverjanje pravočasne prispelosti sporočila. Ta mehanizem zagotavlja, da sporočilo prispe v časovnem okviru 150 sekund. S tem se prepreči možnost zakasnitvenega (*delay*) in ponovitvenega (*replay*) napada [6]. Drugi mehanizem, ki ga uporablja **USM**, je zasebnost z uporabo šifriranja. Ta mehanizem zagotavlja, da pošiljatelj šifrira protokolni podatkovni del paketa **SNMP** (PDU). To stori s privatnim ključem *privKey*. Ko se vzpostavi mehanizem zasebnosti med pošiljateljem in naslovnikom, celoten promet poteka v šifriranem načinu. Za šifriranje in dešifriranje mehanizem uporablja metodo **DES** (angl. *data encryption standard*) [6].

USM tako nudi 3 nivoje varnosti:

- **noAuthNoPriv** nivo ne uporablja ne avtenticiranja ne zasebnosti, kar pomeni, da nudi enak nivo varnosti kot **SNMPv1** in **SNMPv2**;
- **authNoPriv** nivo omogoča avtenticiranje, ne pa tudi šifriranja podatkov;
- **authPriv** nivo zagotavlja avtentikacijo in šifriranje podatkov, kar zagotavlja najbolj varno povezavo.

2.4 Varnostni model VACM

Podsistem za nadzor dostopa protokola **SNMP** je odgovoren za preverjanje ali je dovoljen dostop do določenega objekta za branje in pisanje v bazi **MIB**. Vsak uporabnik, ki želi dostopati do določenih objektov, mora biti definiran v pravi skupini. Različnim skupinam se potem dodeli dostope do različnih nivojev. Tako ima lahko uporabnik, ki pripada določeni skupini le pravice za branje, medtem ko lahko uporabnik, ki pripada skupini z višjim nivojem pravic, spreminja vrednosti objektov v bazi **MIB**. Osnovni gradniki modela **VACM** so [3]:

- skupine (angl. *Groups*),
- raven varnosti (angl. *securityLevels*),
- kontekst (angl. *Contexts*),
- vpogledi **MIB** (angl. *MIB Views*),
- pravilnik dostopov (angl. *Access Policy*).

Postopek varnostnega modela **VACM** odgovarja na sledeča vprašanja [3]:

- Kdo želi dostop?
- Kam želi uporabnik dostopati?
- Kateri varnostni model in raven varnosti, se želi uporabiti?
- Kakšen je namen dostopa do informacij (branje, pisanje ali pošiljanje obvestil)?
- Do katerega objekta želi uporabnik dostopati?

Poglavje 3

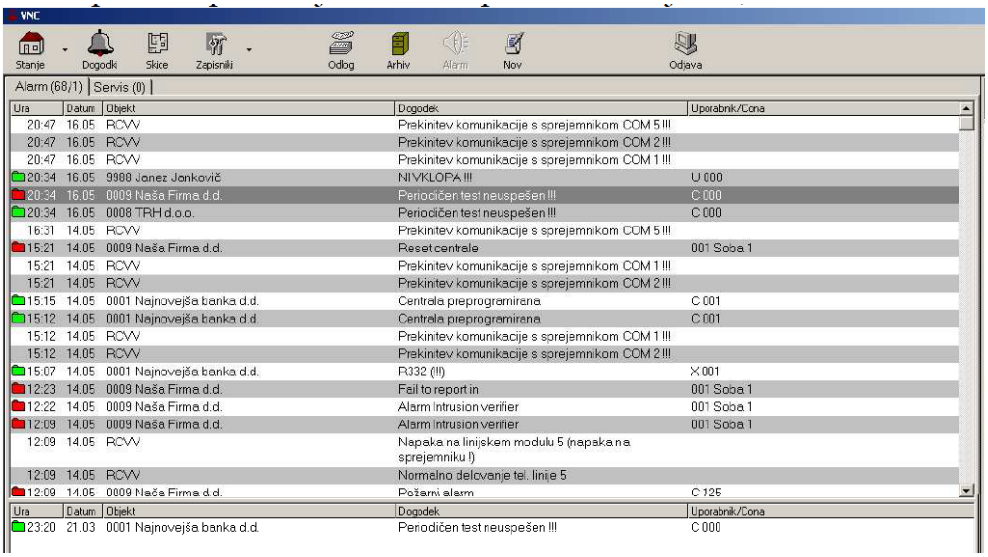
Obstoječe programske rešitve

Na tržišču obstaja veliko različnih rešitev za nadzor opreme in procesov. V podjetju, kjer smo izvedli testno postavitev sistema, ki je opisan v tem diplomskem delu, sta v uporabi dve rešitvi, in sicer Alarm podjeta HSI [18] in PcVue podjetja ARC Informatique [17].

3.1 Alarm

Alarm je programska oprema za nadzor in upravljanje z dogodki v varnostno nadzornih centrih. Zajema obdelavo vhodnih sporočil z strojne opreme, na kateri so lahko nameščeni različni senzorji (temperaturni senzor, senzor odprtosti vrat, dimni senzor). Pridobljene podatke shranjuje v podatkovno bazo Microsoft SQL Server. Pri sprejemu sporočil se večina dogodkov avtomatizirano obdeluje, tako da operaterji dobivajo prikazana le nujna sporočila. Sistem zna ob mejnih dogodkih pošiljati naslovnikom obvestila po eletronski pošti in kratkih tekstovnih sporočilih SMS. Sistem deluje z omejenim številom modelov krmilnikov, prav tako je na krmilnik možno priklopiti le namenske senzorje. Za komunikacijo s strojno opremo je uporabljen protokol SIA IP, ki omogoča kritptiran prenos podatkov. Programska oprema je plačjiva in zaprtokodna. Cena osnovnega paketa za 1000 nadzorovanih objektov je okoli 3500 eur (cena objavljena na internetni strani podjetja), moduli

za razširitev funkcionalnosti se obračunajo dodatno (SMS, modul za nadzor več kot 1000 objektov itd.) [18]. Aplikacija omogoča tudi vnos tlorisa prostora in označitev nadzorovane opreme. Slabe strani sistema glede na naše zahteve so beleženje dogodkov in ne vrednosti temperature, ni vpogleda v zgodovino stanja vrednosti senzorja, slaba združljivost strojne opreme, zaprt in plačljiv sistem. Prikaz aplikacije Alarm je razviden na Sliki 3.1.

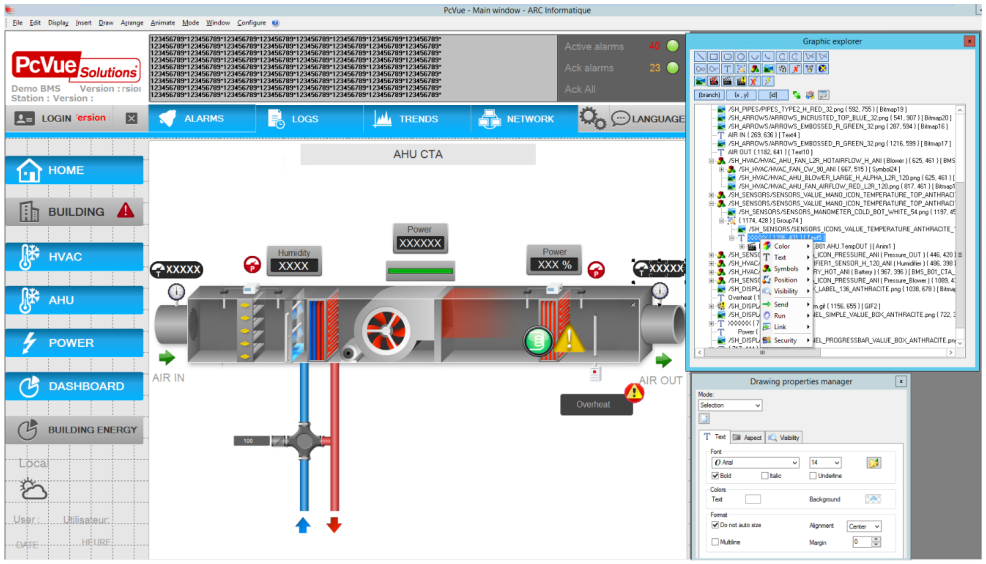


Slika 3.1: Prikaz aplikacije Alarm.

3.2 PcVue

PcVue je orodje **HMI/SCADA** (angl. *Supervisory Control And Data Acquisition*), ki se uporablja za nadzor in krmiljenje industrijskih sistemov. Orodje omogoča alarmiranje ob mejnih dogodkih, nastavljanje parametrov industrijskih sistemov, vpogled v zgodovino in grafično predstavitev stanj. Sistem ima preprost uporabniški vmesnik. Podpira vrsto komunikacijskih protokolov za industrijske standarde (modbus, DNP3, BACnet), standardni internetni protokol za nadzor in upravljanje naprav povezanih v računalniško omrežje (SNMP) itd. Aplikacija shranjuje podatke v bazo SQL Server. Orodje po-

nuja grafične objekte industrijskih elementov, ki jih lahko sestavimo v celoto in tako vizualiziramo naše nadzorovane sisteme (npr. klima naprave, kotle za vročo vodo itd.). Natančne cene niso objavljene na internetu, je pa osnovni paket dražji od 5000 eurov. Prav tako je potrebno za vsako dodatno funkcionalnost dokupiti razširitveni modul (npr. web vpogled do 5 uporabnikov, priklop dodatnih naprav itd.) [17]. Slabe strani sistema glede na naše zahteve so visoka cena osnovnega programa in dodatnih licenc, kompleksna začetna postavitev, zaprt sistem, in slaba združljivost strojne opreme. Slika 3.2 prikazuje vmesnik aplikacije PcVue.



Slika 3.2: Prikaz vmesnika aplikacije PcVue [17].

Poglavje 4

Strojna oprema

V tem poglavju si bomo ogledali strojno opremo, ki smo jo uporabili pri izvedbi naše rešitve.

4.1 Temperaturni senzor DS18B20

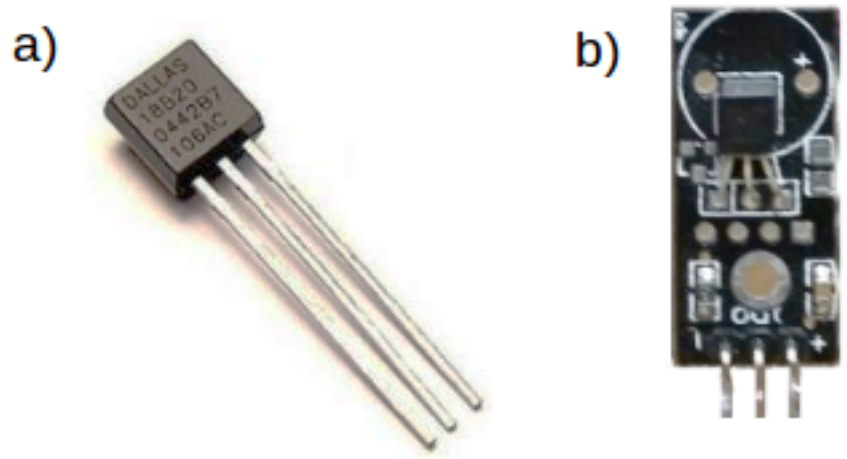
DS18B20 je digitalni temperaturni senzor, ki ga proizvaja podjetje Maxim Integrated Products, Inc. Senzor komunicira po vodilu „1-wire“, katerega glavna prednost je komuniciranje s centralno procesno enoto po eni podatkovni liniji. Prav tako je omogočeno napajanje po podatkovni liniji, zaradi česar ni potrebe po dodatni napajalni enoti. Vsak od senzorjev ima svojo unikatno 64-bitno serijsko kodo, kar omogoča delovanje več senzorjev na enem vodilu hkrati. Senzor lahko izmeri temperaturo v območju od -55 °C do +125 °C. Za nemoteno delovanje potrebuje napetost od -0.5 V do +6.0 V [8].

Zaradi želje po preprosti implementaciji rešitve in enostavnem priklopu na računalnik Raspberry Pi, smo v našem primeru izbrali senzor z že vgrajenim dvigovalnim uporom (angl. *pull-up resistor*). Na Sliki 4.1 sta prikazana:

(a) temperaturni senzor DS18B20, na katerega je potrebno pred uporabo priklopiti še dvigovalni upor;

(b) temperaturni senzor z že vgrajenim dvigovalnim uporom, katerega brez

dodatnih sprememb priklopimo na nožice (*PIN*) računalnika Raspberry Pi.



Slika 4.1: a) Temperaturni senzor DS18B20 brez dvigovalnega upora; b) Temperaturni senzor DS18B20 z vgrajenim dvigovalnim uporom.

Razdalja med uporabljenim računalnikom in priklopljenim senzorjem, je v našem primeru znašala do 10 metrov.

4.2 Računalnik Raspberry Pi

Raspberry Pi je poceni računalnik dimenzij kreditne kartice. Nanj lahko priklopimo osnovne zunanje enote kot so miška, tipkovnica, monitor/televizor. Na nožice lahko priklopimo različne senzorje in naprave. Izdelan je bil z namenom približevanja računalniške tehnologije tako učencem v osnovnih šolah kot tudi starejšim, izkušenim uporabnikom, ki želijo razvijati lastne rešitve [9].

Na tržišču so trenutno na voljo trije modeli Raspberry Pi (A, B in Zero) z različnimi generacijami (1, 1+, 2, 3). Modeli in generacije se med seboj razlikujejo po številu usb priključkov, količini delovnega pomnilnika, po porabi električne energije, številu vhodno-izhodnih signalov (angl. *PIN*) ter

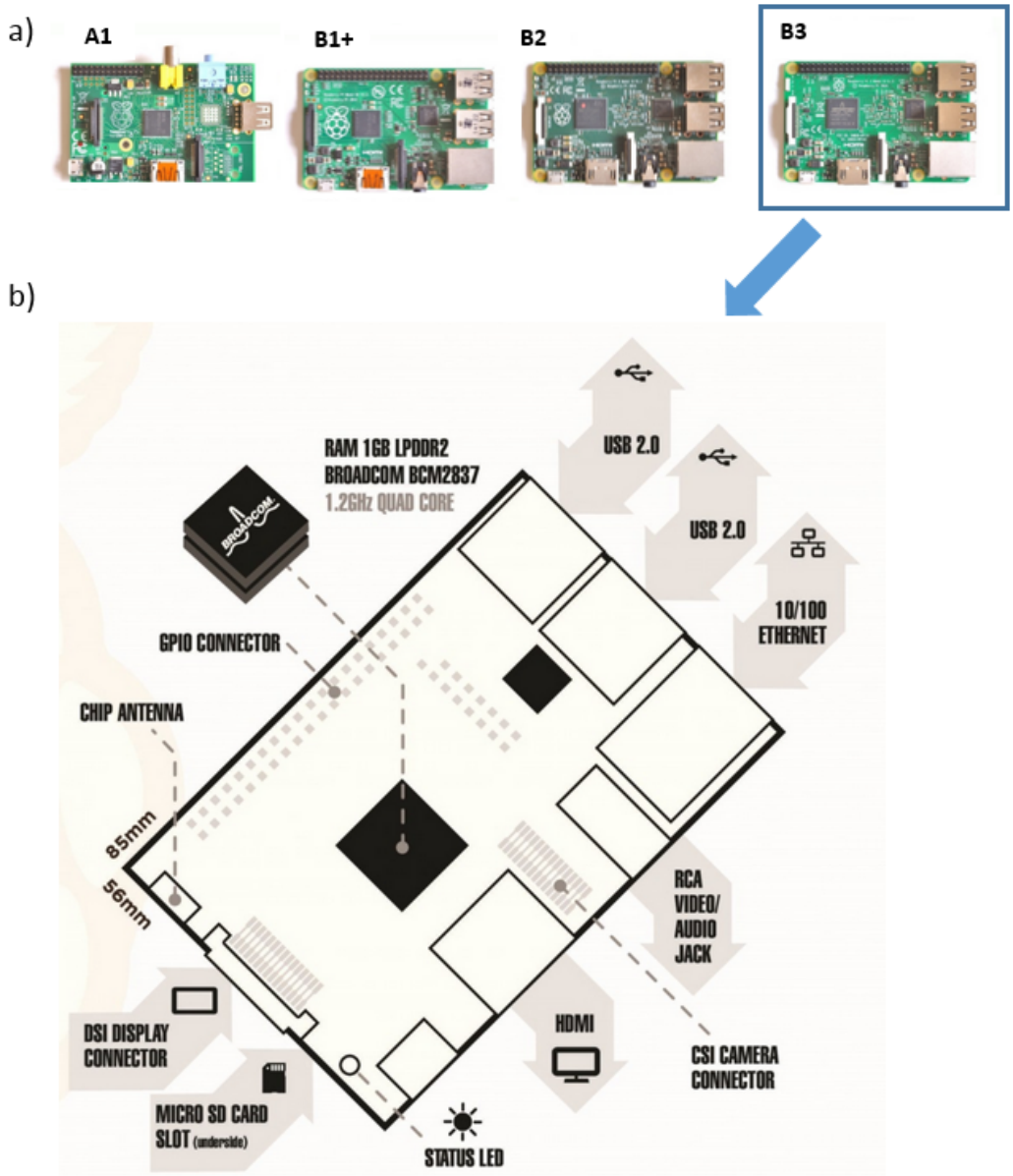
po arhitekturi in hitrosti centralne procesne enote (CPE). Tako se na prvi generaciji nahaja centralna procesna enota z arhitekturo ARM 6 in s frekvenco delovanja 700 MHz, druga generacija ima vgrajen procesor ARM 7 s frekvenco delovanja 900 MHz, procesor na tretji generaciji je ARM 8 s frekvenco delovanja 1,2 GHz. Na enotah se nahaja od 256 MB do 1 GB delovnega pomnnilnika. Na prvi generaciji je za priklop senzorjev na voljo 26 nožic, medtem ko je na novih generacijah teh nožic 40.

Za priklop na računalniško omrežje je na vseh verzijah, z izjemo Raspberry Pi Zero, nameščen IEEE 803.2 ethernet priključek. Ena od večjih prednosti generacije 3 je v tem, da ima poleg mrežnega ethernet priklopa vgrajen tudi brezžični modul IEEE 802.11n. Raspberry Pi za svoje delovanje potrebuje napetost 5 V.

Računalnik za namestitev operacijskega sistema in shranjevanje podatkov uporablja SD kartico (angl. *Secure Digital*). Namestitev operacijskega sistema na kartico je zelo preprosta, ker so na voljo že pripravljene podatkovne slike. Izbiramo lahko med različnimi operacijskimi sistemi kot so GNU/Linux, RISC OS, AROS, ter različnimi GNU/Linux distribucijami:

- Raspbian,
- OpenElec,
- Raspbmc
- Pidora,
- Arch Linux itd.

Izbrali smo GNU/Linux distribucijo Raspbian Jessie, ki je osnovana na distribuciji Debian Jessie. Kot je že omenjeno v samem uvodu, smo želeli postaviti zanesljivo in prilagodljivo infrastrukturo, katera ni zasnovana le na eni verziji računalnika Raspberry Pi. Zaradi tega razloga smo uporabili 5 računalnikov Raspberry Pi različnih inačic. Na Sliki 4.2 so prikazani modeli in generacije, ki smo jih v sklopu diplomskega dela uporabili ter prikazana natančna sestava računalnika Raspberry Pi generacije 3.



Slika 4.2: a) Različni modeli in generacije računalnikov Raspberry Pi, ki smo jih uporabili; b) Opis sestavnih komponent na računalniku Raspberry Pi, modela B, generacije 3 (B3) [9].

Poglavje 5

Programska Oprema

Obstoječi rešitvi (Alarm in PcVue), ki smo ju opisali v razdelku 3, ne ustrezata našim zahtevam, saj smo iskali poceni in odprtokodno rešitev, ki omogoča tudi preprosto razširljivost. Osredotočili smo se na brezplačne nadzorne sisteme, med katerimi so zelo priljubljeni Nagios[1], Zenoss[2] in Zabbix[3]. V našem primeru smo izbrali sistem **Nagios**, katerega glavna prednost je zanesljivost sistema in velika izbira vtičnikov, ki nam olajšajo prikaz podatkov in nadzor nad opremo. Za izrisovanje grafov smo namestili vtičnik **PNP4Nagios**, status temperaturnega senzorja pa smo preverjali s pomočjo vtičnika **check_snmp**, ki je omogočil komunikacijo z računalnikom Raspberry Pi po protokolu **SNMP**. Za implementacijo protokola na agentu **SNMP** smo uporabili paket orodij **Net-SNMP**.

5.1 Net-SNMP

Net-SNMP je zbirka programske opreme, ki jo uporabljamo za implementacijo vseh inačic **SNMP** (SNMPv1, SNMPv2c in SNMPv3). Zbirka vključuje [10]:

[1] https://www.nagios.org/
[2] https://www.zenoss.com/
[3] http://www.zabbix.com/

- razširljivega agenta za odziv na poizvedbe **SNMP** po podatkih za upravljanje (*snmpd*);
- vrsto aplikacij, ki jih izvajamo v ukazni vrstici za:
  - pridobivanje informacij iz naprave preko protokola **SNMP** s posamezno zahtevo (*snmpget*, *snmpgetnext*) ali več zahtevami (*snmpwalk*, *snmptable*, *snmpdelta*),
  - nastavitev parametrov na napravi, ki ima nameščenega agenta **SNMP** (*snmpset*),
  - pridobivanje zbirke podatkov iz naprav z nameščenim agentom **SNMP** (*snmpdf*, *snmpnetstat*, *snmpsatus*),
  - pretvorbo numerične oblike identifikatorja **OID** v tekstualno in za prikaz vsebine in strukture baze **MIB** (*snmptranslate*),
- prikriti proces za spremljanje obvestil **SNMP** (*snmptrapd*);
- grafični brskalnik baze **MIB** (*tkmib*).

Zbirko programske opreme **Net-SNMP** smo namestili na računalnike Raspberry Pi.

5.2 Nadzorni sistem Nagios

Nagios je odprtokodni in brezplačen nadzorni sistem, ki ustreza industrijskim standardom. Namenjen je spremljanju stanja mrežne opreme, mrežnih protokolov, aplikacij, storitev, strežnikov in ostale računalniške opreme. Prednosti, poleg samega nadzora, so [11]:

- Opozorilni sistem, ki pošilja opozorilna sporočila ob preddoločenih dogodkih, ki jih določi administrator. Opozorila lahko sistem pošlje preko elektronske pošte, sporočil SMS ali ukaznih datotek.

- Sistem za poročanje, ki prikazuje zgodovino zapisov vseh obvestil, izpadov sistemov, dogodkov in opozoril. Na podlagi takšnih poročil lahko načrtujemo nadgradnjo sistemov, še preden pride do izpada le-teh.
- Odzivanje na opozorila in dogodke, katere lahko sistem posreduje na različne skupine v primeru, da se primarna skupina ne odzove na dogodke.
- V primeru načrtovanega izvajanja vzdrževalnih del lahko izklopimo nadzor nad določeno opremo in tako začasno preprečimo pošiljanje opozoril.
- Poskus samodejne odprave težave. Upravitelj dogodkov poskuša preko izbirne sistemske ukazne datoteke odpraviti težavo, še preden se sproži dogodek za obveščanje naslovnikov.

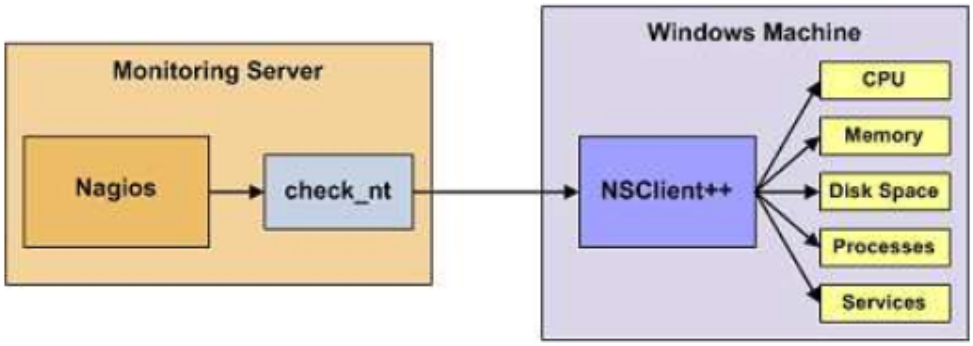
Da lahko sistem **Nagios** nadzira storitve, je potrebno na gostitelja namestiti agenta. Če bi želeli nadzorovati storitve, ki tečejo na operacijskem sistemu Windows, bi to lahko storili z namestitvijo agenta NSClient++ ali NC_Net na gostitelja. Agent se tako obnaša kot namestnik (*proxy*) med Nagios vtičnikom *check_nt* in storitvijo, ki teče na sistemu Windows [12]. Primer uporabe agenta NSClient++ je grafično prikazan na Sliki 5.1. Podobno velja tudi za nadzor sitemov Linux/Unix ter za nadzor opreme kot so tiskalniki, mrežna oprema in ostalo.

V našem primeru smo za dostop do podatkov na gostitelju uporabili protokol **SNMP**. Za komuniciranje s programi **Net-SNMP** smo uporabili vtičnik **check_snmp**. Le-ta pridobi preko poizvedbe **SNMP** informacijo o stanju naprave in jo posreduje nadzornemu sistemu **Nagios**. Opisana zgradba sistema je prikazan na Sliki 5.2.

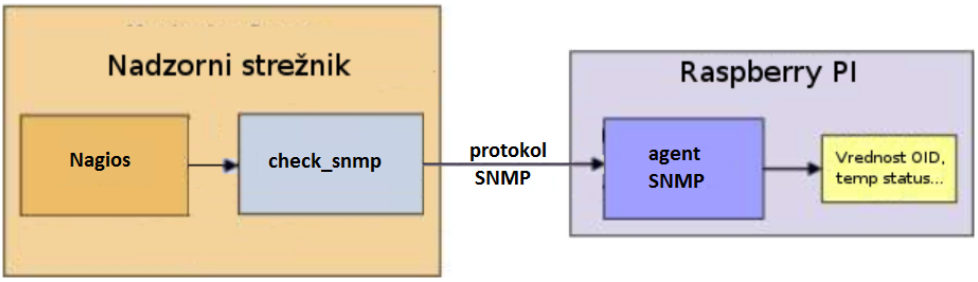
Ena od močnih strani sistema **Nagios** so vtičniki. Veliko uporabnih vtičnikov se namesti že pri osnovni namestitvi sistema, lahko pa po potrebi namestimo tudi dodatne oziroma napišemo vtičnik kar sami.

Da bi lahko pričeli z nadzorom gostitelja, je potrebno v sistemu **Nagios** nastaviti osnove gradnike [13]:

- **Gostitelj (*Host*):** predstavlja definicijo opreme ali aplikacije, ki jo želimo nadzorovati.
- **Storitev (*Service*):** parameter na gostitelju, ki ga želimo nadzorovati.
- **Kontakti (*Contacts*):** nabor naslovnikov, katerim nadzorni sistem **Nagios** pošilja sporočila ob določenih dogodkih.



Slika 5.1: Primer nadzora računalnika z nameščenim operacijskim sistemom Windows [12].



Slika 5.2: Prikaz zgradbe nadzornega sistema **Nagios**, ki preko vtičnika **net_snmp** komunicira po protokolu **SNMP** [12].

Definicijo gostitelja in storitev, ki jih želimo nadzorovati na nadzornem

strežniku (operacijski sistem Linux), nastavimo v datoteki */usr/local/nagios/etc/objects/ime_gostitelja.cfg*.

Primer osnovne definicije gostitelja z imenom *pitemp1* in IP naslovom 192.168.1.80:

```
define host {
       host_name   pitemp1
       alias       pitemp1
       address     192.168.1.80
}
```

Primer osnovne definicije storitve, ki preverja ali naprava odgovarja na zahtevo PING:

```
define service {
       host_name              pitemp1
       service_description    PING
       check_command          check_ping!100.0,20%500.0,60%
}
```

Poleg osnovnih nastavitev lahko gostitelju in storitvi nastavljamo tudi dodatne parametre, v nasprotnem primeru **Nagios** uporabi privzete nastavitve. Nekaj zelo uporabnih parametrov je opisanih v spodnjem seznamu, povzetem po [13], medtem ko je celoten seznam opisan v dokumentaciji sistema **Nagios** [12]:

max_check_attempts: definiramo, kolikokrat bo **Nagios** sprožil preverjanje stanja, potem ko je zaznal napako. Primer parametra nastavljenega na vrednost 3 pomeni, da bo 3-krat preveril stanje, preden bo postavil stanje storitve ali gostitelja na vrednost DOWN;

check_period: časovni okvir, v katerem bo **Nagios** nadzoroval sistem. Privzeta nastavitev je 24 x 7;

notification_interval: parameter določa, kako pogosto naj sistem preveri

stanje storitve ali gostitelja. Vrednost parametra 60 pomeni, da bo sistem preveril stanje vsako uro;

notification_period: parameter določa, v katerih časovnih okvirih bo **Nagios** pošiljal obvestila v primeru zaznanih težav.

Ko imamo gostitelja in storitve nastavljene, je potrebno konfiguracijske datoteke vpisati v datoteko */usr/local/nagios/etc/nagios.cfg*. Primer zapisa konfiguracijskih datotek je viden na Sliki 5.3, vrstica 9, 10, 11, 12, 13.

```
1  # You can specify individual object config files as shown below:
2  cfg_file=/usr/local/nagios/etc/objects/commands.cfg
3  cfg_file=/usr/local/nagios/etc/objects/contacts.cfg
4  cfg_file=/usr/local/nagios/etc/objects/timeperiods.cfg
5  cfg_file=/usr/local/nagios/etc/objects/templates.cfg
6
7  # Definitions for monitoring the local (Linux) host and Raspberry Pi hosts
8  cfg_file=/usr/local/nagios/etc/objects/localhost.cfg
9  cfg_file=/usr/local/nagios/etc/objects/pitemp1.cfg
10 cfg_file=/usr/local/nagios/etc/objects/pitemp2.cfg
11 cfg_file=/usr/local/nagios/etc/objects/pitemp3.cfg
12 cfg_file=/usr/local/nagios/etc/objects/pitemp4.cfg
13 cfg_file=/usr/local/nagios/etc/objects/pitemp5.cfg
```

Slika 5.3: Prikaz zapisanih konfiguracij za gostitelje Raspberry Pi v datoteki *nagios.cfg*.

Trenutno stanje naprave in njenih storitev je v sistemu **Nagios** določeno z vrednostima [14]:

- status gostitelja ali storitve: vrednosti so lahko OK, WARNING, UP, DOWN, CRITICAL, UNREACHABLE,
- stanje gostitelja ali storitve: vrednosti sta lahko SOFT ali HARD; te vrednosti so zelo pomembne pri mehanizmu sproženja dogodkov *event_handler* ali obveščanja naslovnikov.

Storitev ali naprava je lahko v stanju lažje napake (SOFT state) ali v stanju težje napake (HARD state).

Gostitelj ali storitev preideta v lažje stanje v primeru, ko:

- preverjanje statusa naprave ali storitve vrne vrednost non-OK ali non-UP in ko nismo še dosegli števila ponovnih preverjanj, ki smo jih nastavili v konfiguraciji. Tako stanje se imenuje *stanje lažje napake* (angl. *soft error state*);

- storitev ali naprava okrevata in nista več v stanju napake. Temu rečemo *okrevanje po lažji napaki* (angl. *Soft recovery*).

O težjem stanju govorimo takrat, ko:

- preverjanje statusa naprave ali storitve vrne vrednost non-OK ali non-UP in ko smo dosegli število ponovnih preverjanj. Takemu stanju pravimo *stanje težje napake* (angl. *hard error state*);

- gostitelj ali storitev prehaja iz enega stanje težje napake v drugo stanje težje napake (primer iz statusa WARNING v status CRITICAL);

- preverjanje storitev vrne vrednost non-OK - gostitelj storitve je v statusu DOWN ali UNREACHABLE;

- si gostitelj ali storitev iz stanja težje napake opomoreta. Temu pravimo *okrevanje po težji napaki* (angl. *hard revovery*).

V obeh opisanih primerih statusov se napake zabeležijo in izvedejo se upravitelji dogodkov. V primeru težje napake se dodatno obvesti tudi vse naslovnike, ki so definirani v sistemu **Nagios**.

Shranjevanja in vizualizacije pridobljenih podatkov (iz nadzorovanih gostiteljev ali storitev) v obliki grafov sistem **Nagios Core** v sami osnovi ne omogoča. Za shranjevanje podatkov in prikazovanje grafov smo uporabili dodatek **PNP4Nagios**.

5.3 Sistem PNP4Nagios

PNP4Nagios je dodatek za nadzorni sistem **Nagios**, ki analizira podatke (*performance data*), kateri so posredovani s strani vtičnikov in jih samodejno

shranjuje v podatkovno bazo **RRD** (Round Robin Databases). Za obdelavo pridobljenih podatkov **PNP4Nagios** uporablja več načinov [15]:

- **Sinhroni način** (angl. *Synchronous Mode*): ta način je najlažje nastaviti. Slaba stran je ta, da sistem **Nagios** za vsako storitev ali gostitelja izvede ukazno datoteko *process_perfdata.pl*, kar pomeni več sistemskih klicev in s tem večja obremenjenost operacijskega sistema ter nadzornega sistema **Nagios**.

- **Pomožni način** (angl. *Bulk mode*): sistem **Nagios** shranjuje podatke v datoteko in jih v določenih časovnih intervalih obdela s klicem skripte *process_perfdata.pl*. Ta način izvede manj sistemskih klicev, a je obremenjenost sistema **Nagios** še vedno visoka.

- **Pomožni način z NPCD** (angl. *Bulk mode with NPCD*): ta način je priporočljiv za uporabo sistema **Nagios**. Poleg funkcionalnosti, ki jih omogoča *pomožni način*, je glavna prednost v tem, da prikriti proces NPCD obdeluje podatke in s tem razbremeni delo sistema **Nagios**.

- **Pomožni način z npcdmod** (angl. *Bulk Mode with npcdmod*): enaka funkcionalnost kot pomožni način z NPCD, vendar nekompatibilen z Nagios 4, ki smo ga uporabili v našem primeru.

Vsaka od zgornjih načinov ima svoje prednosti in slabosti, ki se odražajo v načinu in hitrosti obdelave pridobljenih podatkov ter s tem v obremenjenosti samega nadzornega sistema. V našem primeru smo se odločili za uporabo načina **Bulk mode with NPCD**, ki dodatno ne obremenjuje sistema **Nagios** in ne sproža toliko sistemskih klicev kot sinhroni način. V omenjenem načinu **Nagios** shranjuje pridobljene podatke v začasno datoteko in zažene ukaze po preteku določenega časovnega obdobja. Namesto takojšnje obdelave podatkov s skripto *process_perfdata.pl*, je datoteka najprej premaknjena v začasno izhodno mapo (*spool directory*). Prikriti proces **NPCD** (*Nagios Performance C Daemon*) preverja ali so v začasni izhodni mapi datoteke na voljo in če so, le-te posreduje procesu process_*perfdata.pl*. Na ta način

je obdelava podatkov popolnoma ločena od sistema **Nagios** [15]. Delovanje nadzornega sistema **Nagios** in dodatka **PNP4Nagios** v načinu **Bulk mode with NPCD** je prikazano na Sliki 5.4.

Prednosti uporabe takšnega načina so:

- Razbremenitev nadzornega sistema **Nagios** in operacijskega sistema. Obdelava podatkov (*performance data*) je v celoti izveden s strani **NPCD** in tako ostane Nagios-u več časa, da opravlja ostala dela.
- Ni izgub podatkov, saj so vse datoteke shranjene v izhodno mapo. Tudi če pride do nepričakovanega izpada sistema, bo **NPCD** še vedno pridobil podatke in ažuriral bazo **RRD**.

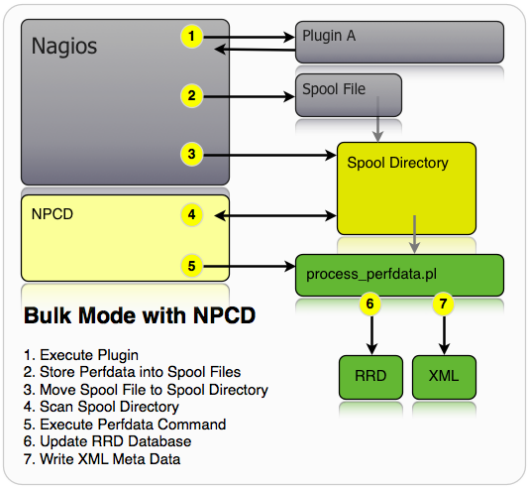
Slabosti takšnega načina:

- Podatki niso obdelani takoj, saj obstaja s strani Nagiosa časovni zamik zapisa podatkov v podatkovno datoteko (*service_perfdata_file_processing_interval*).
- **NPCD** preverja stanje izhodne mape v intervalih po 10 sekund, zato nastane dodatni časovni zamik.

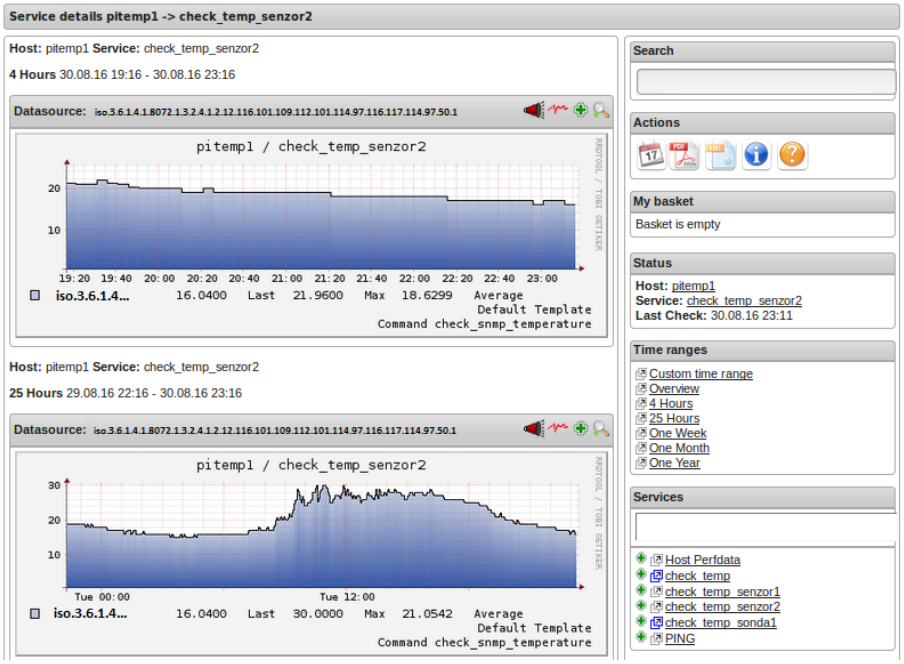
Slika 5.5 prikazuje primer izrisanega grafa stanja temperaturnega senzorja na gostitelju Raspberry Pi, ki ga izrisuje dodatek **PNP4Nagios**. Na prvem grafu so prikazani podatki za zadnje štiri ure in na drugem grafu podatki dnevnega gibanja temperature.

5.4 Orodje RRDtool

Dodatek **PNP4Nagios** za shranjevanje podatkov in izrisovanje grafov uporablja orodje **RRDtool** [16], ki omogoča zaporedno časovno shranjevanje podatkov v podatkovno bazo in na podlagi le-teh grafično predstavitev z grafi. Avtor programske opreme je Tobias Oetiker in orodje je izdano pod licenco GNU General Public License Version 2. Orodje lahko namestimo tako



Slika 5.4: Prikaz delovanja načina Bulk Mode with NPCD [15].



Slika 5.5: Grafični prikaz stanja temperaturnega senzorja na gostitelju Raspberry Pi, ki ga izriše dodatek **PNP4Nagios** [11].

na operacijski sistem Windows kot na operacijske sisteme UNIX. Orodje **RRDtool** ima vmesnike, ki omogočajo uporabo vseh funkcionalnosti v programskih jezikih Python, Pearl, Ruby, Lua, Shell itd. [16].

Osnovni ukazi orodja RRDtool so:

- *create* – tvorjenje podatkovne baze,
- *update* – vnos podatkov v podatkovno bazo,
- *graph* – izris grafov iz podatkov zapisanih v podatkovni bazi.

Ostali ukazi so vidni na Sliki 5.6 v vrsticah 6, 7 in 8.

```
1  mitja@Nagios ~ $ rrdtool
2  RRDtool 1.4.7  Copyright 1997-2012 by Tobias Oetiker <tobi@oetiker.ch>
3                 Compiled Jan  2 2014 19:40:30
4
5  Usage: rrdtool [options] command command_options
6  Valid commands: create, update, updatev, graph, graphv,  dump, restore,
7                  last, lastupdate, first, info, fetch, tune,
8                  resize, xport, flushcached
9
10 RRDtool is distributed under the Terms of the GNU General
11 Public License Version 2. (www.gnu.org/copyleft/gpl.html)
12
13 For more information read the RRD manpages
```

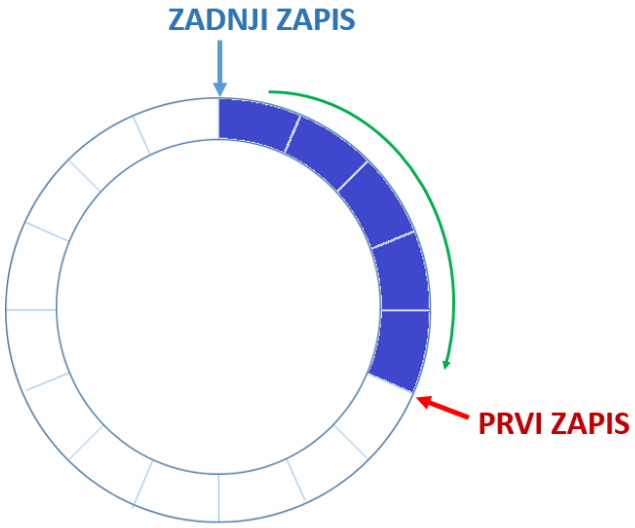
Slika 5.6: Prikaz ukazov orodja RRDtool.

5.4.1 Podatkovna baza orodja RRDtool

RRD je podatkovna baza orodja **RRDtool**. Baza je krožni vmesnik, katerega shranjevanje podatkov si lahko predstavljamo kot krožnico, na kateri posamezne točke predstavljajo podatek (Slika 5.7). Količino podatkov, ki jih želimo zapisati, moramo nastaviti že pri začetni kreaciji baze (ukaz *create* z orodjem **RRDtool**). Ko na krožnici pridemo do prvega zapisa, se le-ta prepiše z novo vrednostjo. Tako ostane velikost baze vedno enaka, kar omogoča preprosto vzdrževanje, saj sistemskemu administratorju ni potrebno skrbeti za povečevanje prostora. Za vsako zapisano vrednost je v krožni vmesnik zapisana tudi časovna oznaka (angl. *time stamp*). Le-ta je

izražena v številu sekund, ki so pretekle od začetka štetja časa UNIX. Če povemo drugače, je to število sekund, ki je preteklo od datuma 1.1.1970.

Orodje **RRDtool** poleg začetne definicije velikosti podatkovne baze, zahteva tudi definicijo preddoločenih časovnih intervalov. Če sistem ne prejme vrednosti v zahtevanem časovnem intervalu, bo avtomatično vpisal vrednost *UNKNOWN*. Zato je zelo pomembno, da s pomočjo skripte pravočasno posredujemo podatke sistemu **RRDtool**.



Slika 5.7: Prikaz shranjevanja podatkov po algoritmu krožnega vmesnika.

5.4.2 Izris grafov

Druga pomembna funkcionalnost orodja **RRDtool** je možnost izrisovanja grafov. Ukaz „graph“ uporablja interni ukaz „fetch“, s katerim prebere vrednosti iz podatkovne baze. Vsak graf lahko prikazuje več podatkovnih virov (*Data Source*). Možen je tudi prikaz vrednosti iz večih podatkovnih baz na enem grafu. Ukaz „graph“ omogoča tudi pretvorbo podatkov (npr. pretvorba iz Kb v Mb). Možna je tudi uporaba logičnih operatorjev kot so „greather than“, „less than“, „if“, „then“, „else“.

Vizualno lahko vrednosti na grafu predstavimo s petimi različnimi oblikami [16]:

AREA – se izraža kot obarvano področje z vrednostmi kot mejami tega področja,

LINE1 – se izraža kot črta,

LINE2 – se izraža kot črta, ki je debeljša od LINE1,

LINE3 – se izraša kot črta, ki je debelješa od LINE2,

STACK – se izraža tudi kot področje, vendar je postavljeno izpred AREA ali LINE1/2/3.

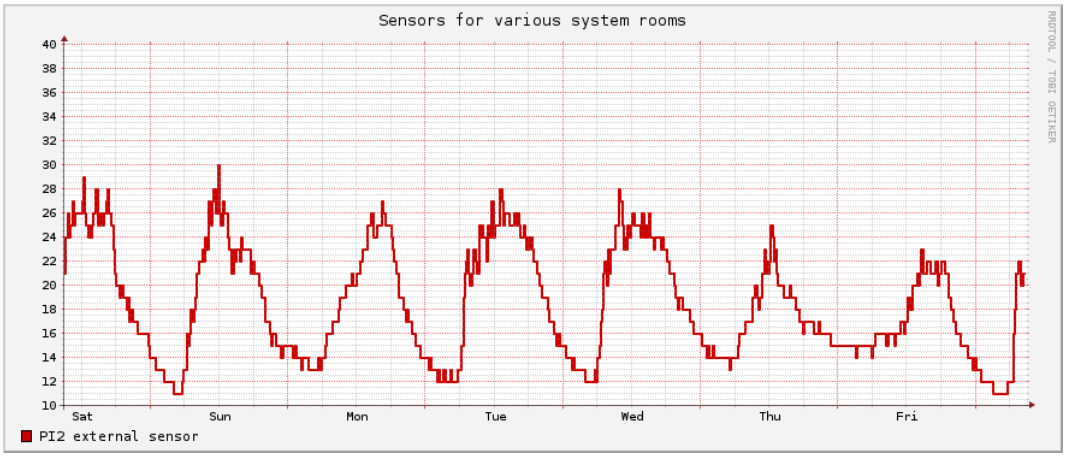
Primer skripte, ki izriše graf s temperaturnimi podatki za obdobje enega tedna:

```
 #!/bin/bash
HOMEDIR=/home/mitja
TEMPDIR=$HOMEDIR/pitemp
DBDIR=$TEMPDIR/data
GRAPHDIR=$HOMEDIR/pitemp
TIMESTAMP=‘date +%s‘
NOW=‘date +$s‘
ONE_WEEK_AGO=$(($NOW-604800))

rrdtool graph $GRAPHDIR/PI2-WEEK.png \
     -- start $ONE_WEEK_AGO --end $NOW -a PNG \
     -- title ‘‘Sensors for various system rooms‘‘ \
     -- width 800 - - height 300 \
     -- color GRID#8c8c8c \
     -- color MGRID#dd1e1e \
```

```
 -- upper-limit 40 - - lower-limit 10 \
DEF:a1=$DBDIR/sensor2.rrd:temp:LAST \
CDEF:b1=a1 \
LINE2:b1#cc0000:"PI2 external sensor"\
```

Izris grafa temperature enega od senzorjev Raspbery Pi z izvedbo zgornje skripte (neposredno z orodjem RRDtool, brez nadzornega sistema Nagios in dodatka PNP4Nagios) je prikazana na Sliki 5.8.



Slika 5.8: Graf prikazuje stanje temperature na enem od naših temperaturnih senzorjev, priklopljenih na računalnik Raspberry Pi. Stanje temperature je prikazano za obdobje enega tedna.

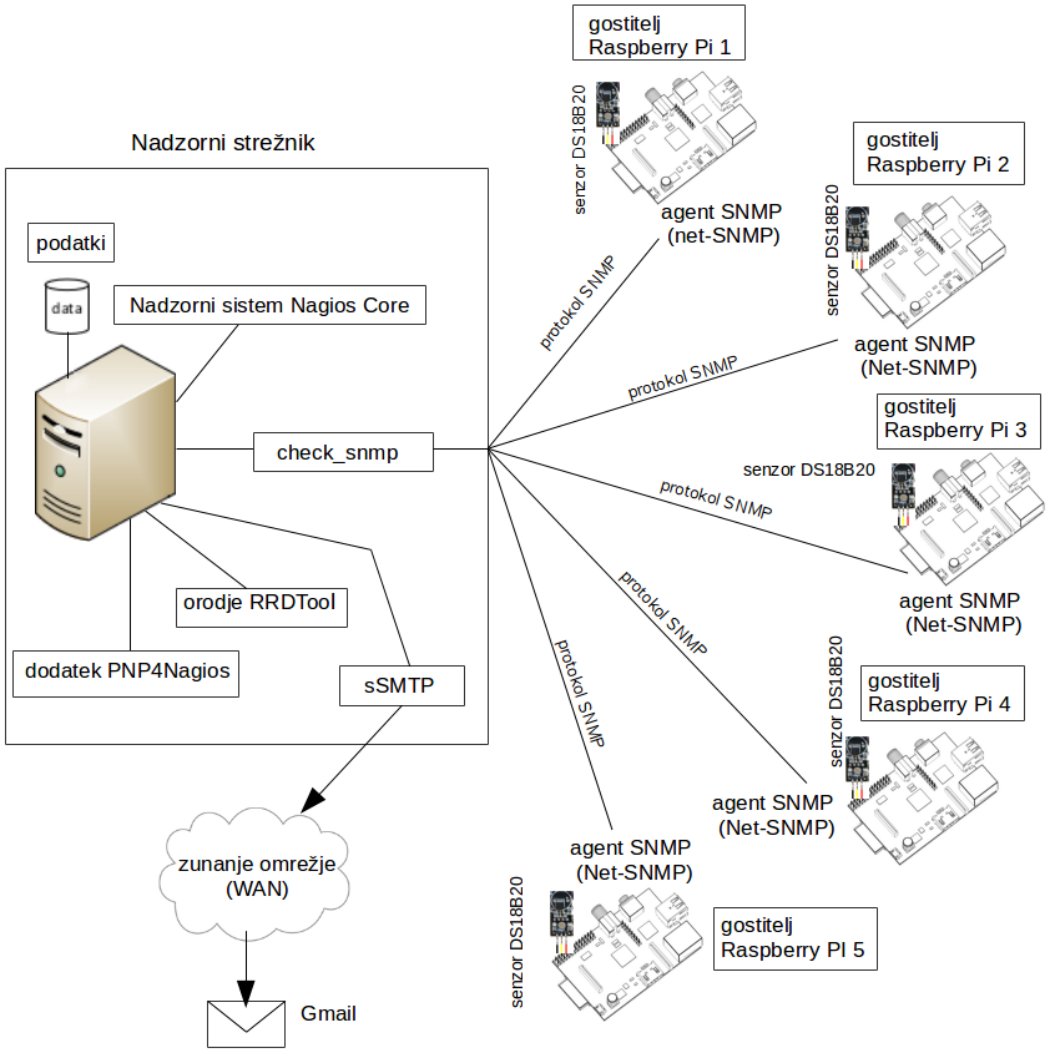
Poglavje 6

Postavitev sistema za nadzor stanja temperaturnih senzorjev

Cilj diplomskega dela je bil postavitev zanesljivega sistema za nadzor temperature v različnih prostorih. Nadzorni sistem preko protokola **SNMP** omogoča prenos podatkov stanja temperaturnih senzorjev z računalnika Rapberry Pi. Na podlagi odstopanj temperature nadzorni sistem sproži obveščanje naslovnikov. Za ta namen smo postavili rešitev s sledečo infrastrukturo:

- strežnik z nameščenim nadzornim sistemom **Nagios Core**, vtičnikom **check_snmp**, dodatkom **PNP4Nagios**, orodjem **RRDTool**, programom **sSMTP** za posredovanje elektronski sporočil uporabnikom,

- 5 računalnikov Raspberry Pi (gostitelji) z nameščenim agentom SNMP (**Net-SNMP**).



Slika 6.1: Prikaz sheme naše implementirane rešitve.

6.1 Priprava računalnika Raspberry Pi

Na računalnik Raspberry Pi smo namestili zadnjo verzijo operacijskega sistema Raspbian Jassie. Temperaturni senzor smo priklopili na sledeče nožice računalnika Raspberry Pi (Slika 6.2):

- ozemljitev (*ground – GND*) na nožico 6 (*Ground*),
- podatkovno vodilo na nožico 7 (*GPIO4*),
- napajanje (*VDD*) na nožico 1 (3.3 V).

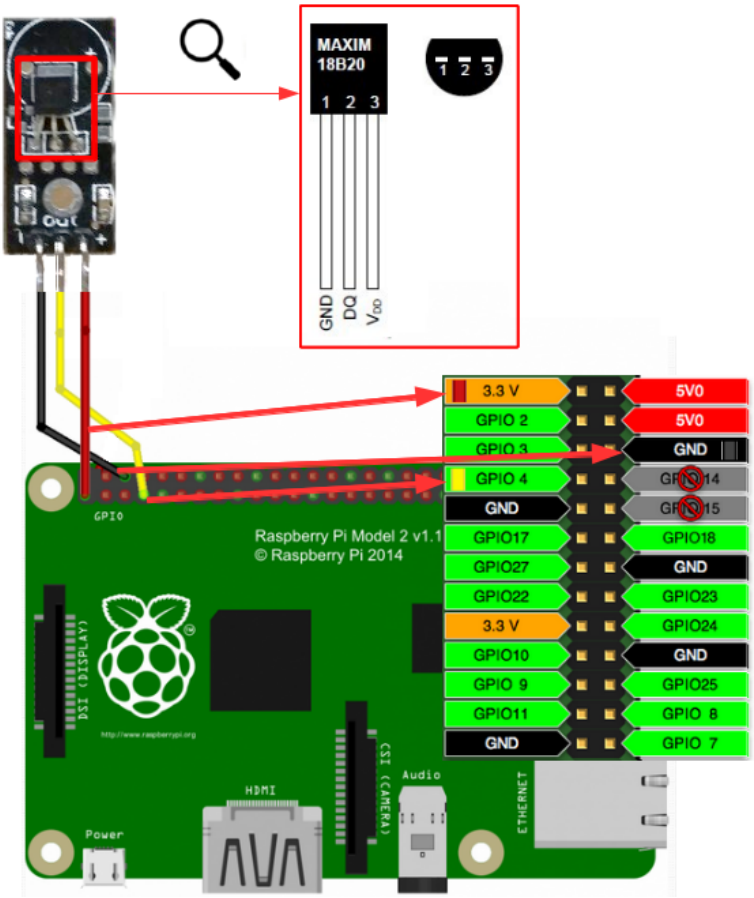
Da bi Raspberry Pi prepoznal temperaturno tipalo, smo morali najprej dodati v konfiguracijsko datoteko */boot/config.txt* niz:

```
dtoverlay=w1-gpio
```

Po spremembi je sistem sam prepoznal senzor in ga namestil v mapo */sys/devices/w1_bus_master1/idštevilkasenzorja*. Mapa z vrednostmi senzorja je bila pri vsakem senzorju drugačna. Pri senzorju z identifikacijsko številko 28-041635538eff lahko vrednost temperature preprosto pridobimo z izpisom datoteke:

```
cat /sys/devices/w1_bus_master1/28-041635538eff/w1_slave
```

Stanje vrednosti temperaturnega senzorja je prikazana na Sliki 6.3. Vrednost temperature znaša 22,250 °C.



Slika 6.2: Prikaz priklopa temperaturnega senzorja DS18B20 na računalnik Raspberry Pi [9, 10].

```
pi@raspberrypi:/sys/devices/w1_bus_master1/28-0416355480ff $ cat w1_slave
6c 01 4b 46 7f ff 0c 10 2b : crc=2b YES
6c 01 4b 46 7f ff 0c 10 2b t=22250
```

Slika 6.3: Prikaz stanja vrednosti temperaturnega senzorja.

Ker je izpis datoteke poleg temperature izpisal za nas nepomembne podatke, smo napisali preprosto skripto v lupini Bash, ki nam izpiše le zaokroženo vrednost temperature:

```
 #!/bin/bash
#pozor:  sensor1 je softlink na datoteko w1_slave
Temp1Probe="/home/pi/pi_temp/sensor1"
Temp1RawFromat=‘cat $Temp1Probe‘
Temperature=‘echo $temp1RawFormat | grep \t=" | cut -d\=
-f3‘
FinalTemp=$((Temperature/1000))
#pozor:  exit zaradi vračanja vrednosti int
exit $FinalTemp
```

Slika 6.4 prikazuje zaokroženo stanje vrednosti temperature, ki je 22 °C.

```
pi@raspberrypi:~/pi_temp $ sudo ./gettemperature.sh
22
pi@raspberrypi:~/pi_temp $
```

Slika 6.4: Izpis stanja temperaturnega senzorja po izvedbi skripte.

Po uspešnem zajemu temperature smo na Raspberry Pi namestili agenta SNMP (orodje **Net-SNMP**). Nadzorni sistem **Nagios** preko vtičnika **check-snmp** in protokola **SNMP** pošlje zahtevo SNMP agentu. Ta agent zatem posreduje zahtevan podatek nazaj vtičniku **check_snmp** (preko protokola **SNMP**)do sistema **Nagios**. Po namestitvi smo v datoteko */etc/snmp/snmp.conf* dodali **SNMP agenta** z:

```
extend temperatura /home/pi/pi_temp/gettemperature.sh
```

Ta ima lastnost *temperatura*, ki jo lahko pridobimo preko skripte *temperature.sh*.

6.1.1 Kloniranje spominske kartice

Ko smo imeli prvi Raspberry Pi dokončno konfiguriran, smo ustvarili sliko (*image*) spominske kartice SD. To smo dosegli z ukazom v lupini Bash

```
dd if=/dev/sdb of=/home/rpiimage.img bs=4096,
```

ki klonira vsebino na kartici (pogon textit/dev/sdb) v datoteko */home/rpiimage.img*. Parameter *bs* (block size) predstavlja število bajtov, ki jih ukaz prebere v bloku in zapiše na spominsko kartico (v našem primeru 4096 bajtov). Kloniranje ostalih spominskih kartic je bilo izvedeno z ukazom

```
dd if=/home/rpiimage.img of=/dev/sdb bs=4096,
```

kjer je */home/rpiimage.img* ime diskovne slike, */dev/sdb* pa predstavlja spominsko kartico SD.

S tem postopkom smo zagotovili zelo preprosto namestitev slike na ostale štiri računalnike Raspberry Pi. Na vsakem računalniku je bilo potrebno spremeniti le številko temperaturnega senzorja, saj ima vsak senzor enolično določeno identifikacijsko številko.

6.1.2 Uporaba Net-SNMP na agentih

Na Raspberry Pi smo namestili programsko opremo **Net-SNMP** in sicer zadnjo različico 5.7.2.1 (Slika 6.5). Omenjena različica omogoča varno komuniciranje med agentom in upravljalcem preko protokola **SNMPv3**. Po uspešni namestitvi programske opreme smo v datoteki */etc/snmp/snmp.conf* popravili sledeče nastavitve:

- privzete nastavitve dovoljujejo le povezavo z lokalnega računalnika, na kateremu je nameščen **Net-SNMP**. Da bi dovolili tudi povezavo našemu nadzornemu **Nagios** strežniku, smo morali dodati vrstico *agentAddress udp:161, udp6:[::1]:161* in odstraniti vrstico *agentAddress udp:127.0.0.1:161*, kar je vidno v vrsticah 17 in 15 na Sliki 6.6.

- zaradi večje varnosti smo uporabili **SNMPv3**. Nastavili smo uporabnika *mitja* in mu določili geslo za generiranje ključev *privKey* ter *authKey*. Prav tako smo nastavili pravice branja poddreves objektov podatkovne baze **MIB** (Slika 6.6, vrstici 37 in 49).

```
pi@raspberrypi:~ $ snmpd --version

NET-SNMP version:  5.7.2.1
Web:               http://www.net-snmp.org/
Email:             net-snmp-coders@lists.sourceforge.net
```

Slika 6.5: Prikaz nameščene verzije programskega paketa **Net-SNMP**.

Net-SNMP dovoljuje razširitev podatkovne baze **MIB** z objektom NET-SNMP-EXTEND-MIB. To storimo tako, da poizvedbo po določeni lastnosti naprave zapišemo v sistemsko ukazno datoteko */etc/snmp/snmp.conf* (Slika 6.7). V konfiguracijski datoteki je najprej potrebno nastaviti sledeče polje:

`ime pot_skripte,`

pri čemer je `ime` niz, po katerem se imenuje novo vozlišče v drevesu podatkovne baze **MIB**, ki ga odpre vsaka pojavitev polja extend. `Pot_skripte` je definicija poti in ime ukazne datoteke, ki jo želimo izvesti. V našem primeru smo tako v datoteko */etc/snmp/snmp.conf* vpisali ukaz

`extend temperatura /home/pi/pi_temp/gettemperature.sh.`

Nastavitev ukaza extend za dva temperaturna senzorja je vidna na Sliki 6.7, vrstici 7 in 8.

S pomočjo ukaza *snmpwalk* naredimo izpis objektov NET-SNMP-EXTEND-MIB::nsExtendResult:

```
snmpwalk -u mitja -l authPriv -a MD5 -x DES -A geslo -X
geslo 192.168.1.80 NET-SNMP-EXTEND-MIB::nsExtendResult
```

Rezultat ukaza je razviden na Sliki 6.8.

Da bi lahko uporabili vtičnik **check_snmp**, ki je nameščen na nadzornem strežniku Nagios, smo morali pridobiti še številko **OID**. Do številke pridemo z ukazom

```
1  ##############################################################################
2  #
3  # EXAMPLE.conf:
4  #   An example configuration file for configuring the Net-SNMP agent ('snmpd')
5  #   See the 'snmpd.conf(5)' man page for details
6  #
7  #  Some entries are deliberately commented out, and will need to be explicitly activated
8  #
9  ##############################################################################
10 #
11 #  AGENT BEHAVIOUR
12 #
13
14 #  Listen for connections from the local system only
15 #agentAddress  udp:127.0.0.1:161
16 #  Listen for connections on all interfaces (both IPv4 *and* IPv6)
17 agentAddress udp:161,udp6:[::1]:161
18
19
20
21 ##############################################################################
22 #
23 #  SNMPv3 AUTHENTICATION
24 #
25 #  Note that these particular settings don't actually belong here.
26 #  They should be copied to the file /var/lib/snmp/snmpd.conf
27 #     and the passwords changed, before being uncommented in that file *only*.
28 #  Then restart the agent
29
30 #  createUser authOnlyUser  MD5 "remember to change this password"
31 #  createUser authPrivUser  SHA "remember to change this one too"  DES
32 #  createUser internalUser  MD5 "this is only ever used internally, but still change the password"
33
34 #  If you also change the usernames (which might be sensible),
35 #  then remember to update the other occurances in this example config file to match.
36
37 createUser mitja MD5 geslo DES
38 #rwuser bootstrap priv
39 rwuser mitja priv
40
41 ##############################################################################
42 #
43 #  ACCESS CONTROL
44 #
45                                            #  system + hrSystem groups only
46 view   systemonly  included   .1.3.6.1.2.1.1
47 view   systemonly  included   .1.3.6.1.2.1.25.1
48
49 included .1.3.6.1.3.1.1
```

Slika 6.6: Nastavitve agenta **SNMP** na Raspberry Pi.

```
1  ##############################################################################
2  #
3  # EXTENDING THE AGENT
4  #
5  # Arbitrary extension commands
6   extend test1 /bin/echo Hello, World!
7   extend temperatura /home/pi/pi_temp/gettemperature.sh
8   extend temperatura2 /home/pi/pi_temp/gettemperature2.sh
```

Slika 6.7: Nastavitev ukazov extend za razširitev podatkovne baze **MIB**. Ukaza temperatura in temperatura2 izpisujeta stanje dveh temperaturnih senzorjev, ki sta nameščena na gostitelju pitemp1.

```
pi@raspberrypi:~ $ snmpwalk -u mitja -l authPriv -a MD5 -x DES -A geslo -X geslo
 192.168.1.80 NET-SNMP-EXTEND-MIB::nsExtendResult
NET-SNMP-EXTEND-MIB::nsExtendResult."test1" = INTEGER: 0
NET-SNMP-EXTEND-MIB::nsExtendResult."test2" = INTEGER: 8960
NET-SNMP-EXTEND-MIB::nsExtendResult."temperatura" = INTEGER: 24
NET-SNMP-EXTEND-MIB::nsExtendResult."temperatura2" = INTEGER: 15
```

Slika 6.8: Prikaz izpisa ukaza snmpwalk za objekte NET-SNMP-EXTEND-MIB::nsExtendResult.

```
snmptranslate -On
NET-SNMP-EXTEND-MIB::nsExtendResult.\"temperatura\"
```

Pridobljena številka **OID** za objekt temperatura, ki smo ga pridobili z ukazom *snmptranslate*, je .1.3.6.1.4.1.8072.1.3.2.4.1.2.11.116.101.109.112.101.114.97.116.117.114.97.1 (Slika 6.9).

```
pi@raspberrypi:~ $ snmptranslate -On NET-SNMP-EXTEND-MIB::nsExtendResult.\"tempe
ratura\"
.1.3.6.1.4.1.8072.1.3.2.4.1.2.11.116.101.109.112.101.114.97.116.117.114.97.1
```

Slika 6.9: Prikaz izpisa **OID** številke za objekt temperatura, ki smo ga izpisali z ukazom snmptranslate.

6.2 Namestitev nadzornega strežnika

Drugi del rešitve je bil postavitev nadzornega strežnika, ki preko protokola **SNMP** pridobi od agenta podatke o temperaturi (Slika 6.1).

Na strežnik smo namestili:

- operacijski sistem Linux Mint 17.3,
- zadnjo verzijo nadzornega sistema **Nagios Core 4.1.1** z osnovnimi vtičniki in zahtevanim paketom **LAMP** (*Linux, Apache, MySQL, PHP*),
- **sSMTP** program za obveščanje uporabnikov na naslov elektronske pošte, ki se nahaja zunaj lokalnega računalniškega omrežja,
- dodatek **PNP4Nagios** za obdelavo temperaturnih podatkov in vizualno predstavitev v orodju **Nagios** in
- orodje **RRDtool** za shranjevanje podatkov o temperaturi v baze **RRD** in izrisovanje grafov.

6.3 Nastavitev nadzornega sistema

Za beleženje vrednosti stanja temperaturnega senzorja na gostitelju Raspberry Pi je bilo potrebno na strani nadzornega sistema **Nagios** definirati:

- ukaz za proženje vtičnika **check_snmp**,
- gostitelje in storitve, ki jih želimo nadzorovati.

Ukaz za proženje vtičnika **check_snmp** smo zapisali v datoteko */usr/local/nagios/etc/objects/commands.cfg*:

```
define command {
    command_name    check_snmp_temperature
    command_line    /usr/lib/nagios/plugins/check_snmp -H
                    $HOSTADRESS$ -P 3 -U mitja -L authPriv
                    -A geslo -X geslo -o $ARG1$ -w  $ARG2$
                    $ARG3$
}
```

Polje *command_name* definira ime ukaza, na katerega se sklicujemo v definiciji storitve. V polju *command_line* definiramo ukaz s parametri. V našem primeru bo sistem **Nagios** ob klicu ukaza *check_snmp_temperature* sprožil vtičnik **check_snmp** skupaj s štirimi dodatnimi parametri:

- *HOSTADRESS*, katerega vrednost bo IP številka agenta, ki je zapisana v definiciji gostitelja (host),
- *ARG1*, katerega vrednost bo **OID** številka, na podlagi katere nam bo gostitelj Raspberry Pi vrnil vrednost temperaturnega senzorja,
- *ARG2*, ki je mejna vrednost temperature, pri kateri bo sistem **Nagios** prestavil storitev iz stanja NORMAL v stanje WARNING ter sprožil obveščanje naslovnikov,
- *ARG3*, ki je mejna vrednost temperature, pri kateri bo nadzorni sistem **Nagios** storitev iz stanja NORMAL ali WARNING prestavil v stanje CRITICAL ter o tem obvestil naslovnike.

Poleg ukaza, ki kliče vtičnik **check_snmp**, smo za vsak Raspberry Pi definirali še gostitelja in storitev v konfiguracijski datotekah */usr/local/nagios/etc/objects/pitempx.cfg*. Primer zapisa definicije gostitelja in storitve Raspberry Pi 1:

```
define host {
        use         linux-server, host-pnp
        host_name   pitemp1
        alias       pitemp1
        address     192.168.1.80
} define service {
        use                   generic-service, srv-pnp
        host_name             pitemp1
        service_description   check_temp_sensor1
        normal_check_interval 5
        notification_interval 30
        check_command         check_snmp_temperature
1.3.6.1.4.1.8072.1.3.2.4.1.2.11.116.101.109.112.101.114.
97.116.117.114.97.1!24!27
}
```

Naš gostitelj Raspberry Pi z imenom pitemp1 (polje *host_name*) in IP številko 192.168.1.80 ima nastavljeno storitev, katerih parametri so:

use generic-service, srv-pnp, s čimer določimo, da želimo shranjevati podatke o temperaturi v podatkovno bazo in izrisovati grafe. To funkcijo opravi sistem **PNP4Nagios**.

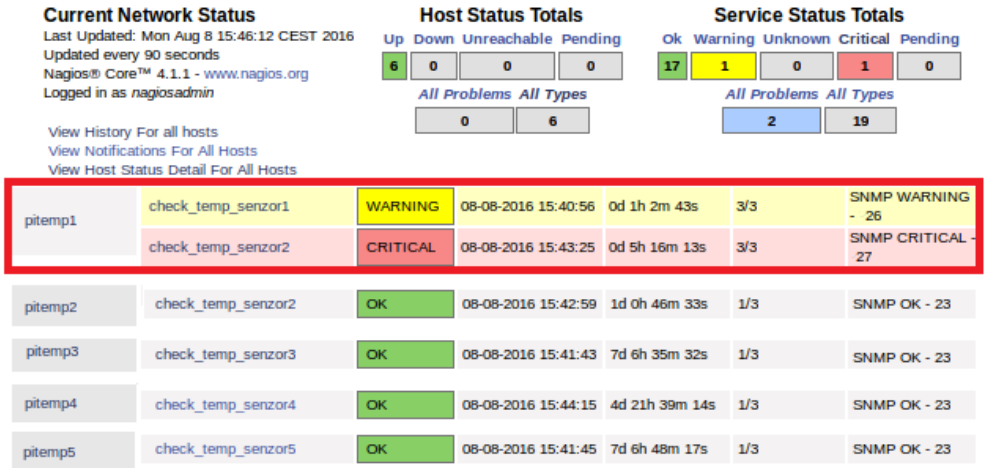
check_command check_snmp_temperature 1.3.6.1.4.1.8072.1.3.2.4.1.2.11.116.101.109.112.101.114.97.116.117.114.97.1!24!27, ki bo sprožil izvedbo vtičnika *check_snmp*. V primeru, da bo zabeležena temperaturna vrednost 24 °C ali več, bo storitev prešla v stanje WARNING. V primeru vrednosti temperature 27 °C ali več, bo storitev prešla v stanje CRITICAL.

normal_check_interval 5, ki določa, da bo sistem **Nagios** storitev preverjal vsakih 5 minut.

notification_interval 30, ki določa, da bodo naslovniki prejemali sporočila

vsakih 30 minut, če bo storitev prešla v stanje WARNING ali CRITICAL.

Primer prikaza stanj petih Rasbperry Pi gostiteljev in storitev v nadzornem sistemu **Nagios** je prikazan na Sliki 6.10.

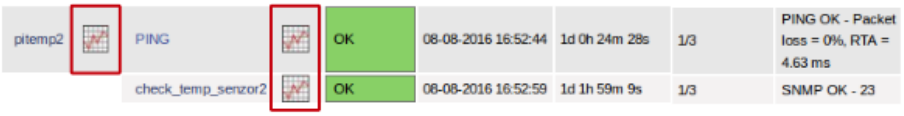


Slika 6.10: Prikaz gostiteljev in storitev v nadzornem sistemu **Nagios**.

Na sliki 6.10 je tudi razvidno, da imamo na gostitelju Raspberry Pi 1 definirani 2 storitvi. Prva storitev je v stanju WARNING, ker je zabeležena temperatura višja od 24 °C. Druga storitev je v stanju CRITICAL, ker je temperatura višja od 27 °C.

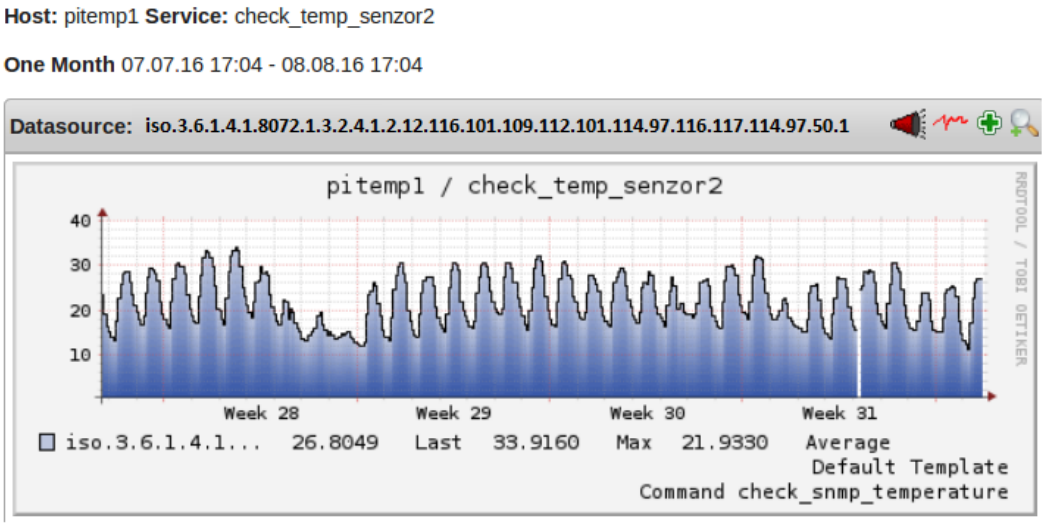
Eden od kriterijev pri snovanju sistema je bila možnost grafičnega prikaza podatkov o temperaturi in možnost vpogleda v zgodovino stanj. To funkcijo opravlja sistem **PNP4Nagios** in sicer tako, da se vrednosti zapišejo v datoteko **XML** in **bazo RRD**. V datoteki **XML** so shranjena sporočila nadzornega sistema, vsi rezultati zapisani v **bazo RRD** pa služijo za izrisovanje grafov. Dodatek **PNP4Nagios** za vsakega od storitev in gostiteljev ustvari svojo **RRD** podatkovno bazo.

Na Sliki 6.11 lahko vidimo, kako vtičnik **PNP4nagios** doda funkcijo izrisovanja grafov.



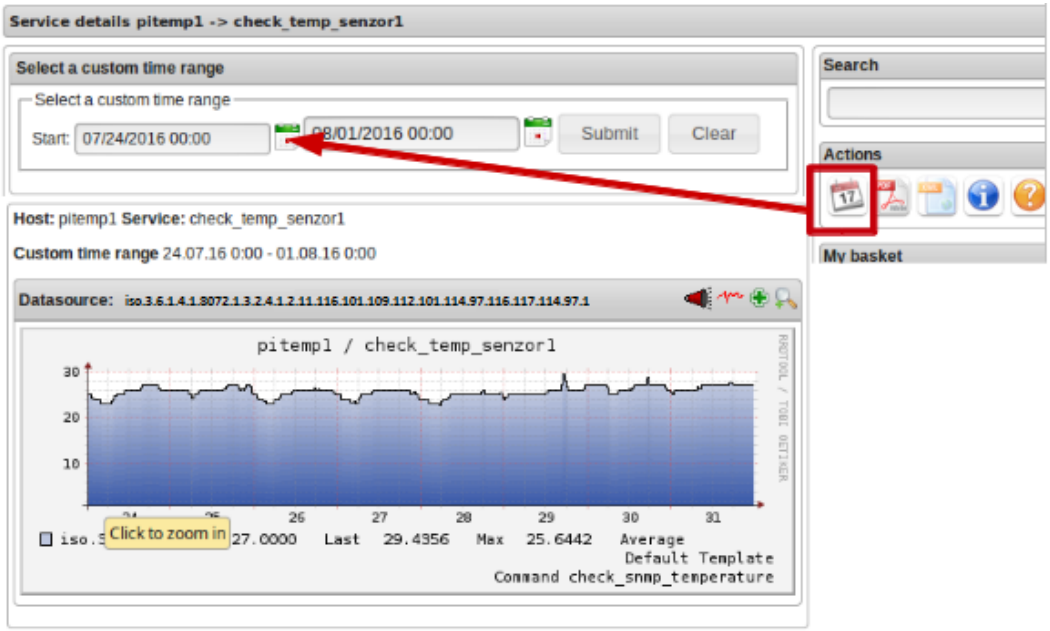
Slika 6.11: Prikaz delovanja vtičnikika **PNP4Nagios**.

Primer izrisa stanja temperature na gostitelju Raspberry Pi 1 in sekundarnem senzorju, ki se nahaja zunaj stavbe, je prikazan na Sliki 6.12. Podatki so prikazani za obdobje enega meseca. V legendi grafa je vidno tudi stanje zadnje, povprečne in najvišje izmerjene temperature.



Slika 6.12: Prikaz stanja temperature za obdobje enega meseca na gostitelju Raspberry Pi 1.

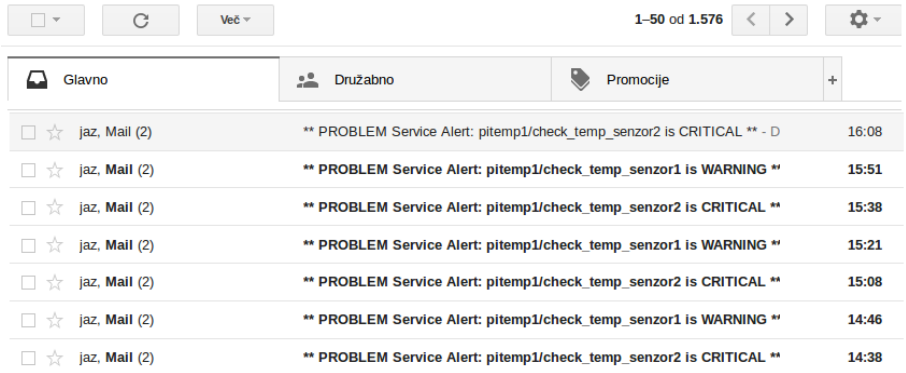
Sistem omogoča tudi vpogled v zgodovino stanj temperaturnega senzorja. Na Sliki 6.13 smo izbrali časovni okvir enega tedna. Rezultat prikazuje stanje temperaturnega senzorja 1 na gostitelju Raspberry Pi 1.



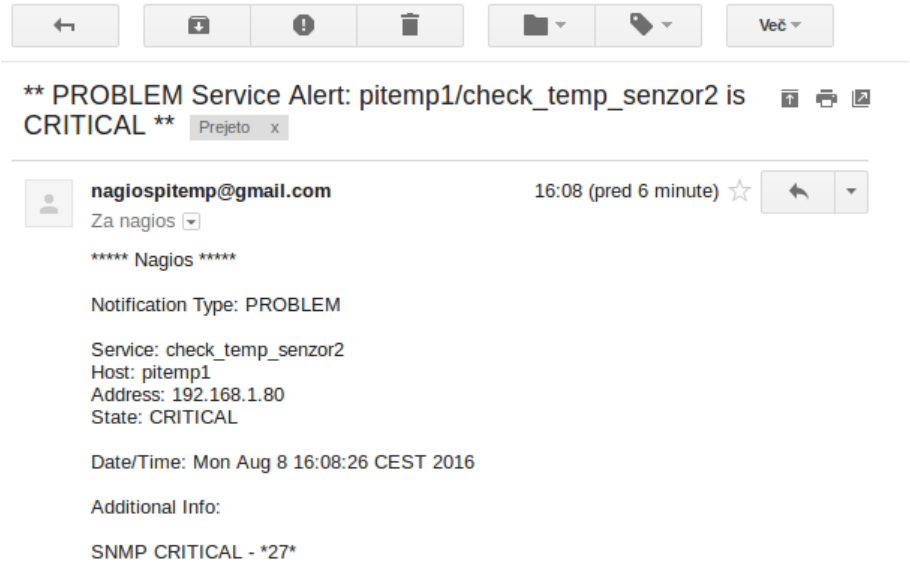
Slika 6.13: Prikaz stanja temperature za preteklo obdobje enega tedna.

6.4 Obveščanje uporabnikov po elektronski pošti

Eden od kriterijev, ki smo ga določili na začetku pisanja diplomskega dela, je funkcija obveščanja uporabnikov po elektronski pošti. Ker smo želeli, da sistem **Nagios** pošlje elektronsko sporočilo o stanju na zunanji elektronski naslov, smo na nadzorni strežnik namestili program **sSNMP**. Program posreduje elektronska sporočila na zunanji poštni strežnik. V našem primeru je to storitev gmail. Na Sliki 6.14 in Sliki 6.15 je vidno, da sistem pošilja obvestila o napakah v časovnem intervalu 30 minut (to vrednost smo nastavili v razdelku 6.3). Vsebina sporočila vsebuje podatek o imenu in IP naslovu gostitelja, stanju storitve in vrednost temperature.



Slika 6.14: Prikaz prejetih elektronskih sporočil, ki jih je posredoval sistem **Nagios** v razmiku 30 minut.



Slika 6.15: Prikaz vsebine prejetega elektronskega sporočila.

Poglavje 7

Zaključek

Za pridobivanje podatkov o temperaturi smo uporabili 5 računalnikov Raspberry Pi, na katere smo priklopili temperaturne senzorje DS18B20. Na nadzorni strežnik smo namestili operacijski sistem Mint in nadzorni sistem **Nagios Core**. Računalnik Raspberry Pi smo definirali kot **agenta SNMP** z namestitvijo skupka orodij **Net-SNMP**. Nadzorni strežnik je preko vtičnika **check_snmp** in **protokola SNMP** s strani gostitelja pridobil podatke o temperaturi in jih s pomočjo **dodatka PNP4Nagios** shranil v **bazo RRD**, hkrati pa poskrbel za izrisovanje grafov. Na koncu smo postavljeno rešitev nadgradili tudi s sistemom, ki ob mejnih dogodkih sproži obveščanje naslovnikov po elektronski pošti. V procesu izdelave diplomskega dela smo postavili rešitev, ki pokriva večino zahtevanih kriterijev, ki smo si jih zadali na samem začetku:

nizka cena in razpoložljivost strojne opreme na trgu: cena Raspberry Pi in temperaturnega senzorja ne presega vrednosti 45 eurov. Strojna oprema je na voljo v vsaki trgovini z elektroniko. Računalnik Raspberry Pi Zero v vrednosti 5 eurov bi našo rešitev še pocenil, žal pa je bil v času pisanja diplomske naloge nedobavljiv. Pomankljivost modela Zero je tudi ta, da nima ethernet priključka.

zanesljiva in poceni rešitev na ravni programske opreme: nadzorni sistem

Nagios Core je zanesljiv in brezplačen sistem, ki ponuja podobne funkcionalnosti kot plačljivi sistemi.

možnost preprostega dodajanja nove opreme in menjave opreme v primeru okvare: v primeru okvare računalnika Raspberry Pi je potrebno le zamenjati strojno opremo in vstaviti kartico SD. V primeru okvare SD kartice je potrebno izvesti postopek kloniranja diskovne slike. Če želimo dodati nov računalnik Raspberry Pi, pripravimo kartico SD in popravimo vrednost identifikacijske številke temperaturnega senzorja.

preprosta administracija: nadzor sistema in dodajanje novih gostiteljev sta preprosta.

obveščanje naslovnikov: sistem omogoča pošiljanje na poljuben e-poštni naslov.

možnost razširljivosti sistema z dodatnimi nadzornimi senzorji: Raspberry Pi omogoča priklop vseh vrst senzorjev kot so senzor za nadzor dima, vlage, nadzor prižganosti luči in ostalo. Na podoben način kot smo v diplomskem delu priklopili temperaturni senzor, lahko dodamo tudi ostale senzorje ter s tem razširimo funkcionalnost sistema.

grafični prikaz podatkov in zgodovine vrednosti stanj: s pomočjo dodatka **PNP4Nagios** smo poskrbeli za tedenski, mesečni in letni prikaz stanja temperature. Prav tako lahko sami izberemo obdobje prikaza podatkov.

Našo rešitev smo primerjali z obstoječima rešitvama, ki smo jih opisali v razdelku 3. Izkazalo se je, da imata rešitvi sledeče pomankljivosti, zaradi katerih je naša rešitev primernejša:

Alarm:

- sistem ne beleži podatkov, ampak beleži le dogodke, ki se sprožijo ob določenih vrednostih (npr. ko temperatura preseže dovoljene vrednosti);

- ne zapisuje vrednosti stanj senzorjev v podatkovno bazo, zaradi česar ni na voljo vpogleda v zgodovino in izrisovanja grafov;
- cena posameznega krmilnika je zelo visoka (podprti so le določeni modeli) in ravno tako vsaka nadgradnja sistema (nove funkcije, podpora novim senzorjem, ...);
- gre za zaprt, nefleksibilen sistem, zaradi česar razvoj znotraj podjetja ni mogoč;
- za pregled dogodkov mora biti na računalnik nameščen odjemalec za urejanje dogodkov (ni spletnega vmesnika);
- programska oprema je plačljiva.

PcVue:

- za nekatere funkcionalnosti je potrebno plačati licence (npr. za upravljanje in vpogled preko spletnega vmesnika);
- potrebna je podpora zunanjega dobavitelja;
- izvorna koda ni dostopna;
- potrebno je priklopiti združljivo strojno opremo (npr. krmilnik Zelio logic) in senzorje, katerih cena je visoka;
- visoka cena programske opreme PcVue.

Iz Tabele 7.1 je razvidno, da je glede na zastavljene kriterije, naša rešitev najbolj primerna.

Kriterij	**Naša rešitev**	**Alarm**	**PcVue**
centralni nadzor	da	da	da
spletni vmesnik	da	ne	da
cena osn. programske opreme	0 eur	3700 eur	>5000 eur
plačljivi dodatni moduli	ne	da	da
grafični prikaz podatkov	da	ne	da
vpogled v zgodovino stanj	da	ne	da
način obveščanje uporabnikov	mail	mail, sms	mail, sms
možnost internega razvoja	da	ne	ne
vrednost strojne opreme na enoto	45 eur	>200 eur	>500 eur
vzdrževanje strojne opreme	nezahtevno	zahtevno	zahtevno

Tabela 7.1: Primerjava rešitev po različnih kriterijih.

Sistem je trenutno postavljen kot ena od mnogih možnosti, ki jih ponuja trg. Trenutno se sistem uporablja v testni fazi. V primeru končne izbire bi se osredotočili na nadaljnji razvoj funkcionalnosti. Sistem bi lahko izboljšali z vizualizacijo nadzorovanih prostorov, kar bi lahko izvedli s pomočjo dodatka NagVis. Prav tako bi dodali še nadzor nad dodatnimi senzorji (npr. senzor dima) in obveščanje naslovnikov s sporočili SMS. Z dodatnimi funkcijami bi tako še povečali uporabnost sistema in s tem podali dodatne razloge za končno uporabo v produkciji.

Literatura

[1] William Stallings. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Addison-Wesley, 1999.

[2] Douglas R. Mauro, Kevin J. Schmidt. *Essential SNMP, Second Edtition*. O'Reilly Media, September 2005.

[3] Jian Ren, Tongtong Li. *Network Management*. Michigan State University. Dostopno na: http://www.egr.msu.edu/ renjian/pubs/network-management.pdf. Dostopano: 15.07.2016.

[4] Polona Antončič. Monitoriranje računalniških omrežij. Diplomsko delo, Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, 2012.

[5] Larry Walsh. *SNMP MIB Handbook*. Wyndham Press, 2008.

[6] William Stallings. *Security Comes to SNMP: The New SNMPv3 Proposed Internet Standards – The Internet Protocol Journal – Volume 1, No. 3*. Cisco Press. Dostopno na: http://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-20/snmpv3.html. Pregledano: 15.07.2016.

[7] P. Chatzimisios. *Security issues and vulnerabilities of the SNMP protocol*. School of Design, Engineering and Computing, Bournemouth University, UK. Dostopno na: http://www.ee.ucl.ac.uk/lcs/previous/LCS2003/102.pdf. Dostopano: 12.07.2016.

[8] Maxim Integrated. *Application Note 4377.* Dosegljivo na: https://www.maximintegrated.com/en/app-notes/index.mvp/id/4377. Dostopano: 20.07.2016.

[9] Raspberry Pi foundation. *Raspberry Pi products.* Dostopno na: https://www.raspberrypi.org/products/. Dostopano 22.07.2016.

[10] Alex Burger.
textitNet-SNMP. Dostopno na: http://www.net-snmp.org/. Dostopano: 08.07.2016.

[11] Nagios team. *Nagios Core.* Dosegljivo na: https://www.nagios.com/products/nagios-core. Dostopano: 07.07.2016.

[12] Nagios team. *Nagios Documentation.* Dosegljivo na: http://library.nagios.com/library/products/nagioscore/manuals/. Dostopano: 08.07.2016.

[13]] Tom Ryder. *Nagios Core Administration Cookbook, Second Edition, Februar 2016.* Packt publishing, 2016.

[14] Nagios team. *Nagios State Types.* Dosegljivo na: https://assets.nagios.com/downloads
/nagioscore/docs/nagioscore/3/en/statetypes.html. Dostopano: 20.07.2016.

[15] PNP4Nagios team. *PNP4Nagios 0.6 Documentation.* Dosegljivo na: https://docs.pnp4nagios.org/pnp-0.6/start. Dostopano: 21.07.2016.

[16] Tobias Oetiker. *RRDtool.* Dosegljivo na: http://oss.oetiker.ch/rrdtool/. Dostopano: 03.07.2016.

[17] Arcinfo. *PcVue features.* Dosegljivo na: http://www.arcinfo.com/index.php/products-a-technology/pcvue-hmiscada/scada-features. Dostopano: 22.07.2016.

[18] HSI. *Programska oprema Alarm.* Dosegljivo na: http://hsi.si/. Dostopano: 22.07.2016.